TEXAS INSTRUMENTS-PRODUCTION DATA

Tiva™ TM4C1232C3PM Microcontroller

DATA SHEET

DS-TM4C1232C3PM-15842.2741
SPMS341E

Copyright

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

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

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

Table of Contents

Revision History 32

About This Document 35

Audience 35

About This Manual 35

Related Documents 35

Documentation Conventions 36

1 Architectural Overview 38

1.1 Tiva™ C Series Overview 38

1.2 TM4C1232C3PM Microcontroller Overview 39

1.3 TM4C1232C3PM Microcontroller Features 41

1.3.1 ARM Cortex-M4F Processor Core 41

1.3.2 On-Chip Memory 43

1.3.3 Serial Communications Peripherals 45

1.3.4 System Integration 49

1.3.5 Analog 54

1.3.6 JTAG and ARM Serial Wire Debug 56

1.3.7 Packaging and Temperature 56

1.4 TM4C1232C3PM Microcontroller Hardware Details 56

1.5 Kits 57

1.6 Support Information 57

2 The Cortex-M4F Processor 58

2.1 Block Diagram 59

2.2 Overview 60

2.2.1 System-Level Interface 60

2.2.2 Integrated Configurable Debug 60

2.2.3 Trace Port Interface Unit (TPIU) 61

2.2.4 Cortex-M4F System Component Details 61

2.3 Programming Model 62

2.3.1 Processor Mode and Privilege Levels for Software Execution 62

2.3.2 Stacks 63

2.3.3 Register Map 63

2.3.4 Register Descriptions 65

2.3.5 Exceptions and Interrupts 81

2.3.6 Data Types 81

2.4 Memory Model 81

2.4.1 Memory Regions, Types and Attributes 83

2.4.2 Memory System Ordering of Memory Accesses 84

2.4.3 Behavior of Memory Accesses 84

2.4.4 Software Ordering of Memory Accesses 85

2.4.5 Bit-Banding 86

2.4.6 Data Storage 88

2.4.7 Synchronization Primitives 89

2.5 Exception Model 90

2.5.1 Exception States 91

2.5.2 Exception Types 91

2.5.3 Exception Handlers 95
2.5.4 Vector Table 95
2.5.5 Exception Priorities 96
2.5.6 Interrupt Priority Grouping 97
2.5.7 Exception Entry and Return 97
2.6 Fault Handling 100
2.6.1 Fault Types 101
2.6.2 Fault Escalation and Hard Faults 101
2.6.3 Fault Status Registers and Fault Address Registers 102
2.6.4 Lockup 102
2.7 Power Management 103
2.7.1 Entering Sleep Modes 103
2.7.2 Wake Up from Sleep Mode 103
2.8 Instruction Set Summary 104

3 Cortex-M4 Peripherals 111
3.1 Functional Description 111
3.1.1 System Timer (SysTick) 112
3.1.2 Nested Vectored Interrupt Controller (NVIC) 113
3.1.3 System Control Block (SCB) 114
3.1.4 Memory Protection Unit (MPU) 114
3.1.5 Floating-Point Unit (FPU) 119
3.2 Register Map 123
3.3 System Timer (SysTick) Register Descriptions 126
3.4 NVIC Register Descriptions 130
3.5 System Control Block (SCB) Register Descriptions 145
3.6 Memory Protection Unit (MPU) Register Descriptions 174
3.7 Floating-Point Unit (FPU) Register Descriptions 183

4 JTAG Interface 189
4.1 Block Diagram 190
4.2 Signal Description 190
4.3 Functional Description 191
4.3.1 JTAG Interface Pins 191
4.3.2 JTAG TAP Controller 193
4.3.3 Shift Registers 193
4.3.4 Operational Considerations 194
4.4 Initialization and Configuration 196
4.5 Register Descriptions 197
4.5.1 Instruction Register (IR) 197
4.5.2 Data Registers 199

5 System Control 201
5.1 Signal Description 201
5.2 Functional Description 201
5.2.1 Device Identification 201
5.2.2 Reset Control 202
5.2.3 Non-Maskable Interrupt 207
5.2.4 Power Control 207
5.2.5 Clock Control 208
5.2.6 System Control 215

5.3 Initialization and Configuration 219
5.4 Register Map 219
5.5 System Control Register Descriptions 224
5.6 System Control Legacy Register Descriptions 391

6 System Exception Module 449
6.1 Functional Description 449
6.2 Register Map 449
6.3 Register Descriptions 449

7 Internal Memory 457
7.1 Block Diagram 457
7.2 Functional Description 458
7.2.1 SRAM 458
7.2.2 ROM 459
7.2.3 Flash Memory 461
7.2.4 EEPROM 467
7.3 Register Map 472
7.4 Flash Memory Register Descriptions (Flash Control Offset) 474
7.5 EEPROM Register Descriptions (EEPROM Offset) 492
7.6 Memory Register Descriptions (System Control Offset) 509

8 Micro Direct Memory Access (μDMA) 517
8.1 Block Diagram 518
8.2 Functional Description 518
8.2.1 Channel Assignments 519
8.2.2 Priority 520
8.2.3 Arbitration Size 520
8.2.4 Request Types 520
8.2.5 Channel Configuration 521
8.2.6 Transfer Modes 523
8.2.7 Transfer Size and Increment 531
8.2.8 Peripheral Interface 531
8.2.9 Software Request 531
8.2.10 Interrupts and Errors 532
8.3 Initialization and Configuration 532
8.3.1 Module Initialization 532
8.3.2 Configuring a Memory-to-Memory Transfer 533
8.3.3 Configuring a Peripheral for Simple Transmit 534
8.3.4 Configuring a Peripheral for Ping-Pong Receive 536
8.3.5 Configuring Channel Assignments 538
8.4 Register Map 538
8.5 μDMA Channel Control Structure 540
8.6 μDMA Register Descriptions 547

9 General-Purpose Input/Outputs (GPIOs) 581
9.1 Signal Description 581
9.2 Functional Description 584
9.2.1 Data Control 585
9.2.2 Interrupt Control 586
9.2.3 Mode Control 587

9.2.4 Commit Control 588
9.2.5 Pad Control 588
9.2.6 Identification 588
9.3 Initialization and Configuration 588
9.4 Register Map 590
9.5 Register Descriptions 593

10 General-Purpose Timers 636

10.1 Block Diagram 637
10.2 Signal Description 638
10.3 Functional Description 639
10.3.1 GPTM Reset Conditions 640
10.3.2 Timer Modes 641
10.3.3 Wait-for-Trigger Mode 650
10.3.4 Synchronizing GP Timer Blocks 651
10.3.5 DMA Operation 652
10.3.6 Accessing Concatenated 16/32-Bit GPTM Register Values 652
10.3.7 Accessing Concatenated 32/64-Bit Wide GPTM Register Values 652
10.4 Initialization and Configuration 654
10.4.1 One-Shot/Periodic Timer Mode 654
10.4.2 Real-Time Clock (RTC) Mode 655
10.4.3 Input Edge-Count Mode 655
10.4.4 Input Edge Time Mode 656
10.4.5 PWM Mode 656
10.5 Register Map 657
10.6 Register Descriptions 658

11 Watchdog Timers 706

11.1 Block Diagram 707
11.2 Functional Description 707
11.2.1 Register Access Timing 708
11.3 Initialization and Configuration 708
11.4 Register Map 708
11.5 Register Descriptions 709

12 Analog-to-Digital Converter (ADC) 731

12.1 Block Diagram 732
12.2 Signal Description 733
12.3 Functional Description 734
12.3.1 Sample Sequencers 734
12.3.2 Module Control 735
12.3.3 Hardware Sample Averaging Circuit 738
12.3.4 Analog-to-Digital Converter 739
12.3.5 Differential Sampling 742
12.3.6 Internal Temperature Sensor 744
12.3.7 Digital Comparator Unit 745
12.4 Initialization and Configuration 749
12.4.1 Module Initialization 749
12.4.2 Sample Sequencer Configuration 750
12.5 Register Map 750
12.6 Register Descriptions 752

13 Universal Asynchronous Receivers/Transmitters (UARTs) 822
13.1 Block Diagram 823
13.2 Signal Description 823
13.3 Functional Description 824
13.3.1 Transmit/Receive Logic 824
13.3.2 Baud-Rate Generation 825
13.3.3 Data Transmission 826
13.3.4 Serial IR (SIR) 826
13.3.5 ISO 7816 Support 827
13.3.6 Modem Handshake Support 828
13.3.7 9-Bit UART Mode 829
13.3.8 FIFO Operation 829
13.3.9 Interrupts 829
13.3.10 Loopback Operation 830
13.3.11 DMA Operation 831
13.4 Initialization and Configuration 831
13.5 Register Map 832
13.6 Register Descriptions 834

14 Synchronous Serial Interface (SSI) 881
14.1 Block Diagram 882
14.2 Signal Description 882
14.3 Functional Description 883
14.3.1 Bit Rate Generation 883
14.3.2 FIFO Operation 884
14.3.3 Interrupts 884
14.3.4 Frame Formats 885
14.3.5 DMA Operation 893
14.4 Initialization and Configuration 894
14.5 Register Map 896
14.6 Register Descriptions 897

15 Inter-Integrated Circuit (I^{2}C) Interface 926
15.1 Block Diagram 927
15.2 Signal Description 927
15.3 Functional Description 928
15.3.1 I^{2}C Bus Functional Overview 928
15.3.2 Available Speed Modes 933
15.3.3 Interrupts 935
15.3.4 Loopback Operation 936
15.3.5 Command Sequence Flow Charts 936
15.4 Initialization and Configuration 944
15.4.1 Configure the I^{2}C Module to Transmit a Single Byte as a Master 944
15.4.2 Configure the I^{2}C Master to High Speed Mode 945
15.5 Register Map 946
15.6 Register Descriptions (I^{2}C Master) 947
15.7 Register Descriptions (I^{2}C Slave) 964
15.8 Register Descriptions (I^{2}C Status and Control) 974

16 Controller Area Network (CAN) Module 977
16.1 Block Diagram 978
16.2 Signal Description 978
16.3 Functional Description 979
16.3.1 Initialization 980
16.3.2 Operation 980
16.3.3 Transmitting Message Objects 981
16.3.4 Configuring a Transmit Message Object 982
16.3.5 Updating a Transmit Message Object 983
16.3.6 Accepting Received Message Objects 983
16.3.7 Receiving a Data Frame 984
16.3.8 Receiving a Remote Frame 984
16.3.9 Receive/Transmit Priority 985
16.3.10 Configuring a Receive Message Object 985
16.3.11 Handling of Received Message Objects 986
16.3.12 Handling of Interrupts 988
16.3.13 Test Mode 989
16.3.14 Bit Timing Configuration Error Considerations 991
16.3.15 Bit Time and Bit Rate 991
16.3.16 Calculating the Bit Timing Parameters 993
16.4 Register Map 996
16.5 CAN Register Descriptions 997

17 Universal Serial Bus (USB) Controller 1027
17.1 Block Diagram 1027
17.2 Signal Description 1028
17.3 Functional Description 1028
17.3.1 Operation 1028
17.3.2 DMA Operation 1033
17.4 Initialization and Configuration 1034
17.4.1 Endpoint Configuration 1034
17.5 Register Map 1035
17.6 Register Descriptions 1038

18 Analog Comparators 1083
18.1 Block Diagram 1084
18.2 Signal Description 1084
18.3 Functional Description 1085
18.3.1 Internal Reference Programming 1086
18.4 Initialization and Configuration 1088
18.5 Register Map 1088
18.6 Register Descriptions 1089

19 Pin Diagram 1098

20 Signal Tables 1099
20.1 Signals by Pin Number 1100
20.2 Signals by Signal Name 1105
20.3 Signals by Function, Except for GPIO 1111
20.4 GPIO Pins and Alternate Functions 1115
20.5 Possible Pin Assignments for Alternate Functions 1118

20.6 Connections for Unused Signals ... 1120

21 Electrical Characteristics ... 1122

21.1 Maximum Ratings ... 1122

21.2 Operating Characteristics ... 1123

21.3 Recommended Operating Conditions ... 1124

21.4 Load Conditions ... 1126

21.5 JTAG and Boundary Scan ... 1127

21.6 Power and Brown-Out ... 1129

21.6.1 VDDA Levels ... 1129

21.6.2 VDD Levels ... 1130

21.6.3 VDDC Levels ... 1131

21.6.4 VDD Glitches ... 1132

21.6.5 VDD Droop Response ... 1132

21.7 Reset ... 1134

21.8 On-Chip Low Drop-Out (LDO) Regulator ... 1137

21.9 Clocks ... 1138

21.9.1 PLL Specifications ... 1138

21.9.2 PIOSC Specifications ... 1139

21.9.3 Low-Frequency Internal Oscillator (LFIOSC) Specifications ... 1139

21.9.4 Main Oscillator Specifications ... 1139

21.9.5 System Clock Specification with ADC Operation ... 1143

21.9.6 System Clock Specification with USB Operation ... 1143

21.10 Sleep Modes ... 1144

21.11 Flash Memory and EEPROM ... 1146

21.12 Input/Output Pin Characteristics ... 1147

21.12.1 GPIO Module Characteristics ... 1147

21.12.2 Types of I/O Pins and ESD Protection ... 1147

21.13 Analog-to-Digital Converter (ADC) ... 1151

21.14 Synchronous Serial Interface (SSI) ... 1154

21.15 Inter-Integrated Circuit (I^2C) Interface ... 1157

21.16 Universal Serial Bus (USB) Controller ... 1158

21.17 Analog Comparator ... 1159

21.18 Current Consumption ... 1161

A Package Information ... 1164

A.1 Orderable Devices ... 1164

A.2 Device Nomenclature ... 1164

A.3 Device Markings ... 1165

A.4 Packaging Diagram ... 1166

List of Figures

Figure 1-1. Tiva™ TM4C1232C3PM Microcontroller High-Level Block Diagram 40
Figure 2-1. CPU Block Diagram 60
Figure 2-2. TPIU Block Diagram 61
Figure 2-3. Cortex-M4F Register Set 64
Figure 2-4. Bit-Band Mapping 88
Figure 2-5. Data Storage 89
Figure 2-6. Vector Table 96
Figure 2-7. Exception Stack Frame 99
Figure 3-1. SRD Use Example 117
Figure 3-2. FPU Register Bank 120
Figure 4-1. JTAG Module Block Diagram 190
Figure 4-2. Test Access Port State Machine 193
Figure 4-3. IDCODE Register Format 199
Figure 4-4. BYPASS Register Format 199
Figure 4-5. Boundary Scan Register Format 200
Figure 5-1. Basic $\overline{\text{RST}}$ Configuration 204
Figure 5-2. External Circuitry to Extend Power-On Reset 204
Figure 5-3. Reset Circuit Controlled by Switch 205
Figure 5-4. Power Architecture 208
Figure 5-5. Main Clock Tree 210
Figure 5-6. Module Clock Selection 217
Figure 7-1. Internal Memory Block Diagram 457
Figure 7-2. EEPROM Block Diagram 458
Figure 8-1. μDMA Block Diagram 518
Figure 8-2. Example of Ping-Pong μDMA Transaction 524
Figure 8-3. Memory Scatter-Gather, Setup and Configuration 526
Figure 8-4. Memory Scatter-Gather, μDMA Copy Sequence 527
Figure 8-5. Peripheral Scatter-Gather, Setup and Configuration 529
Figure 8-6. Peripheral Scatter-Gather, μDMA Copy Sequence 530
Figure 9-1. Digital I/O Pads 584
Figure 9-2. Analog/Digital I/O Pads 585
Figure 9-3. GPIODATA Write Example 586
Figure 9-4. GPIODATA Read Example 586
Figure 10-1. GPTM Module Block Diagram 637
Figure 10-2. Reading the RTC Value 644
Figure 10-3. Input Edge-Count Mode Example, Counting Down 646
Figure 10-4. 16-Bit Input Edge-Time Mode Example 647
Figure 10-5. 16-Bit PWM Mode Example 649
Figure 10-6. CCP Output, GPTMTnMATCHR > GPTMTnILR 649
Figure 10-7. CCP Output, GPTMTnMATCHR = GPTMTnILR 650
Figure 10-8. CCP Output, GPTMTnILR > GPTMTnMATCHR 650
Figure 10-9. Timer Daisy Chain 651
Figure 11-1. WDT Module Block Diagram 707
Figure 12-1. Implementation of Two ADC Blocks 732
Figure 12-2. ADC Module Block Diagram 733
Figure 12-3. ADC Sample Phases 736

Figure 12-4. Doubling the ADC Sample Rate 737
Figure 12-5. Skewed Sampling 737
Figure 12-6. Sample Averaging Example 739
Figure 12-7. ADC Input Equivalency 740
Figure 12-8. ADC Voltage Reference 741
Figure 12-9. ADC Conversion Result 742
Figure 12-10. Differential Voltage Representation 744
Figure 12-11. Internal Temperature Sensor Characteristic 745
Figure 12-12. Low-Band Operation (CIC=0x0) 747
Figure 12-13. Mid-Band Operation (CIC=0x1) 748
Figure 12-14. High-Band Operation (CIC=0x3) 749
Figure 13-1. UART Module Block Diagram 823
Figure 13-2. UART Character Frame 825
Figure 13-3. IrDA Data Modulation 827
Figure 14-1. SSI Module Block Diagram 882
Figure 14-2. TI Synchronous Serial Frame Format (Single Transfer) 886
Figure 14-3. TI Synchronous Serial Frame Format (Continuous Transfer) 887
Figure 14-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0 888
Figure 14-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0 888
Figure 14-6. Freescale SPI Frame Format with SPO=0 and SPH=1 889
Figure 14-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0 890
Figure 14-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0 890
Figure 14-9. Freescale SPI Frame Format with SPO=1 and SPH=1 891
Figure 14-10. MICROWIRE Frame Format (Single Frame) 892
Figure 14-11. MICROWIRE Frame Format (Continuous Transfer) 893
Figure 14-12. MICROWIRE Frame Format, SSInFss Input Setup and Hold Requirements 893
Figure 15-1. I^{2}C Block Diagram 927
Figure 15-2. I^{2}C Bus Configuration 928
Figure 15-3. START and STOP Conditions 929
Figure 15-4. Complete Data Transfer with a 7-Bit Address 929
Figure 15-5. R/S Bit in First Byte 930
Figure 15-6. Data Validity During Bit Transfer on the I^{2}C Bus 930
Figure 15-7. High-Speed Data Format 935
Figure 15-8. Master Single TRANSMIT 937
Figure 15-9. Master Single RECEIVE 938
Figure 15-10. Master TRANSMIT of Multiple Data Bytes 939
Figure 15-11. Master RECEIVE of Multiple Data Bytes 940
Figure 15-12. Master RECEIVE with Repeated START after Master TRANSMIT 941
Figure 15-13. Master TRANSMIT with Repeated START after Master RECEIVE 942
Figure 15-14. Standard High Speed Mode Master Transmit 943
Figure 15-15. Slave Command Sequence 944
Figure 16-1. CAN Controller Block Diagram 978
Figure 16-2. CAN Data/Remote Frame 979
Figure 16-3. Message Objects in a FIFO Buffer 988
Figure 16-4. CAN Bit Time 992
Figure 17-1. USB Module Block Diagram 1027
Figure 18-1. Analog Comparator Module Block Diagram 1084
Figure 18-2. Structure of Comparator Unit 1085

Figure 18-3. Comparator Internal Reference Structure 1086

Figure 19-1. 64-Pin LQFP Package Pin Diagram 1098

Figure 21-1. Load Conditions 1126

Figure 21-2. JTAG Test Clock Input Timing 1127

Figure 21-3. JTAG Test Access Port (TAP) Timing 1128

Figure 21-4. Power Assertions versus VDDA Levels 1130

Figure 21-5. Power and Brown-Out Assertions versus VDD Levels 1131

Figure 21-6. POK assertion vs VDDC 1132

Figure 21-7. POR-BOR0-BOR1 VDD Glitch Response 1132

Figure 21-8. POR-BOR0-BOR1 VDD Droop Response 1133

Figure 21-9. Digital Power-On Reset Timing 1134

Figure 21-10. Brown-Out Reset Timing 1135

Figure 21-11. External Reset Timing ($\overline{\text{RST}}$) 1135

Figure 21-12. Software Reset Timing 1135

Figure 21-13. Watchdog Reset Timing 1135

Figure 21-14. MOSC Failure Reset Timing 1136

Figure 21-15. ESD Protection on Fail-Safe Pins 1148

Figure 21-16. ESD Protection on Non-Fail-Safe Pins 1149

Figure 21-17. ADC Input Equivalency Diagram 1153

Figure 21-18. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement 1155

Figure 21-19. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer 1155

Figure 21-20. Master Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1 1156

Figure 21-21. Slave Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1 1156

Figure 21-22. I^{2}C Timing 1157

Figure A-1. Key to Part Numbers 1164

Figure A-2. TM4C1232C3PM 64-Pin LQFP Package Diagram 1166

List of Tables

Table 1. Revision History 32
Table 2. Documentation Conventions 36
Table 1-1. TM4C1232C3PM Microcontroller Features 39
Table 2-1. Summary of Processor Mode, Privilege Level, and Stack Use 63
Table 2-2. Processor Register Map 64
Table 2-3. PSR Register Combinations 70
Table 2-4. Memory Map 81
Table 2-5. Memory Access Behavior 84
Table 2-6. SRAM Memory Bit-Banding Regions 86
Table 2-7. Peripheral Memory Bit-Banding Regions 86
Table 2-8. Exception Types 92
Table 2-9. Interrupts 93
Table 2-10. Exception Return Behavior 100
Table 2-11. Faults 101
Table 2-12. Fault Status and Fault Address Registers 102
Table 2-13. Cortex-M4F Instruction Summary 104
Table 3-1. Core Peripheral Register Regions 111
Table 3-2. Memory Attributes Summary 115
Table 3-3. TEX, S, C, and B Bit Field Encoding 117
Table 3-4. Cache Policy for Memory Attribute Encoding 118
Table 3-5. AP Bit Field Encoding 118
Table 3-6. Memory Region Attributes for Tiva™ C Series Microcontrollers 119
Table 3-7. QNaN and SNaN Handling 122
Table 3-8. Peripherals Register Map 123
Table 3-9. Interrupt Priority Levels 153
Table 3-10. Example SIZE Field Values 181
Table 4-1. JTAG_SWD_SWO Signals (64LQFP) 190
Table 4-2. JTAG Port Pins State after Power-On Reset or $\overline{\text{RST}}$ assertion 191
Table 4-3. JTAG Instruction Register Commands 197
Table 5-1. System Control & Clocks Signals (64LQFP) 201
Table 5-2. Reset Sources 202
Table 5-3. Clock Source Options 209
Table 5-4. Possible System Clock Frequencies Using the SYSDIV Field 211
Table 5-5. Examples of Possible System Clock Frequencies Using the SYSDIV2 Field 212
Table 5-6. Examples of Possible System Clock Frequencies with DIV400=1 212
Table 5-7. System Control Register Map 219
Table 5-8. RCC2 Fields that Override RCC Fields 247
Table 6-1. System Exception Register Map 449
Table 7-1. Flash Memory Protection Policy Combinations 462
Table 7-2. User-Programmable Flash Memory Resident Registers 466
Table 7-3. Flash Register Map 473
Table 8-1. μDMA Channel Assignments 519
Table 8-2. Request Type Support 521
Table 8-3. Control Structure Memory Map 522
Table 8-4. Channel Control Structure 522
Table 8-5. μDMA Read Example: 8-Bit Peripheral 531

Table 8-6. μDMA Interrupt Assignments 532
Table 8-7. Channel Control Structure Offsets for Channel 30 533
Table 8-8. Channel Control Word Configuration for Memory Transfer Example 534
Table 8-9. Channel Control Structure Offsets for Channel 7 535
Table 8-10. Channel Control Word Configuration for Peripheral Transmit Example 535
Table 8-11. Primary and Alternate Channel Control Structure Offsets for Channel 8 536
Table 8-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example 537
Table 8-13. μDMA Register Map 539
Table 9-1. GPIO Pins With Special Considerations 582
Table 9-2. GPIO Pins and Alternate Functions (64LQFP) 582
Table 9-3. GPIO Pad Configuration Examples 589
Table 9-4. GPIO Interrupt Configuration Example 590
Table 9-5. GPIO Pins With Special Considerations 591
Table 9-6. GPIO Register Map 592
Table 9-7. GPIO Pins With Special Considerations 603
Table 9-8. GPIO Pins With Special Considerations 609
Table 9-9. GPIO Pins With Special Considerations 611
Table 9-10. GPIO Pins With Special Considerations 614
Table 9-11. GPIO Pins With Special Considerations 620
Table 10-1. Available CCP Pins 638
Table 10-2. General-Purpose Timers Signals (64LQFP) 638
Table 10-3. General-Purpose Timer Capabilities 640
Table 10-4. Counter Values When the Timer is Enabled in Periodic or One-Shot Modes 641
Table 10-5. 16-Bit Timer With Prescaler Configurations 642
Table 10-6. 32-Bit Timer (configured in 32/64-bit mode) With Prescaler Configurations 643
Table 10-7. Counter Values When the Timer is Enabled in RTC Mode 643
Table 10-8. Counter Values When the Timer is Enabled in Input Edge-Count Mode 645
Table 10-9. Counter Values When the Timer is Enabled in Input Event-Count Mode 646
Table 10-10. Counter Values When the Timer is Enabled in PWM Mode 648
Table 10-11. Timeout Actions for GPTM Modes 651
Table 10-12. Timers Register Map 658
Table 11-1. Watchdog Timers Register Map 709
Table 12-1. ADC Signals (64LQFP) 733
Table 12-2. Samples and FIFO Depth of Sequencers 734
Table 12-3. Differential Sampling Pairs 742
Table 12-4. ADC Register Map 750
Table 13-1. UART Signals (64LQFP) 824
Table 13-2. Flow Control Mode 828
Table 13-3. UART Register Map 833
Table 14-1. SSI Signals (64LQFP) 883
Table 14-2. SSI Register Map 896
Table 15-1. I2C Signals (64LQFP) 927
Table 15-2. Examples of I^{2}C Master Timer Period Versus Speed Mode 933
Table 15-3. Examples of I^{2}C Master Timer Period in High-Speed Mode 934
Table 15-4. Inter-Integrated Circuit (I^{2}C) Interface Register Map 946
Table 15-5. Write Field Decoding for I2CMCS[3:0] Field 952
Table 16-1. Controller Area Network Signals (64LQFP) 979

Table 16-2. Message Object Configurations 984
Table 16-3. CAN Protocol Ranges 992
Table 16-4. CANBIT Register Values 992
Table 16-5. CAN Register Map 996
Table 17-1. USB Signals (64LQFP) 1028
Table 17-2. Remainder (MAXLOAD/4) 1033
Table 17-3. Actual Bytes Read 1034
Table 17-4. Packet Sizes That Clear RXRDY 1034
Table 17-5. Universal Serial Bus (USB) Controller Register Map 1035
Table 18-1. Analog Comparators Signals (64LQFP) 1084
Table 18-2. Internal Reference Voltage and ACREFCTL Field Values 1086
Table 18-3. Analog Comparator Voltage Reference Characteristics, V_{DDA} = 3.3V, EN= 1, and RNG = 0 1087
Table 18-4. Analog Comparator Voltage Reference Characteristics, V_{DDA} = 3.3V, EN= 1, and RNG = 1 1087
Table 18-5. Analog Comparators Register Map 1088
Table 20-1. GPIO Pins With Special Considerations 1099
Table 20-2. Signals by Pin Number 1100
Table 20-3. Signals by Signal Name 1105
Table 20-4. Signals by Function, Except for GPIO 1111
Table 20-5. GPIO Pins and Alternate Functions 1115
Table 20-6. Possible Pin Assignments for Alternate Functions 1118
Table 20-7. Connections for Unused Signals (64-Pin LQFP) 1120
Table 21-1. Absolute Maximum Ratings 1122
Table 21-2. ESD Absolute Maximum Ratings 1122
Table 21-3. Temperature Characteristics 1123
Table 21-4. Thermal Characteristics 1123
Table 21-5. Recommended DC Operating Conditions 1124
Table 21-6. Recommended GPIO Pad Operating Conditions 1124
Table 21-7. GPIO Current Restrictions 1124
Table 21-8. GPIO Package Side Assignments 1125
Table 21-9. JTAG Characteristics 1127
Table 21-10. Power-On and Brown-Out Levels 1129
Table 21-11. Reset Characteristics 1134
Table 21-12. LDO Regulator Characteristics 1137
Table 21-13. Phase Locked Loop (PLL) Characteristics 1138
Table 21-14. Actual PLL Frequency 1138
Table 21-15. PIOSC Clock Characteristics 1139
Table 21-16. Low-Frequency internal Oscillator Characteristics 1139
Table 21-17. Main Oscillator Input Characteristics 1139
Table 21-18. Crystal Parameters 1141
Table 21-19. Supported MOSC Crystal Frequencies 1142
Table 21-20. System Clock Characteristics with ADC Operation 1143
Table 21-21. System Clock Characteristics with USB Operation 1143
Table 21-22. Sleep Modes AC Characteristics 1144
Table 21-23. Time to Wake with Respect to Low-Power Modes 1144
Table 21-24. Flash Memory Characteristics 1146
Table 21-25. EEPROM Characteristics 1146

Table 21-26. GPIO Module Characteristics 1147
Table 21-27. Pad Voltage/Current Characteristics for Fail-Safe Pins 1148
Table 21-28. Fail-Safe GPIOs that Require an External Pull-up 1149
Table 21-29. Non-Fail-Safe I/O Pad Voltage/Current Characteristics 1149
Table 21-30. ADC Electrical Characteristics 1151
Table 21-31. SSI Characteristics 1154
Table 21-32. I^2C Characteristics 1157
Table 21-33. Analog Comparator Characteristics 1159
Table 21-34. Analog Comparator Voltage Reference Characteristics 1159
Table 21-35. Analog Comparator Voltage Reference Characteristics, V_{DDA} = 3.3V, EN= 1, and RNG = 0 1159
Table 21-36. Analog Comparator Voltage Reference Characteristics, V_{DDA} = 3.3V, EN= 1, and RNG = 1 1160
Table 21-37. Current Consumption 1161

List of Registers

The Cortex-M4F Processor

The Cortex-M4F Processor 58
Register 1: Cortex General-Purpose Register 0 (R0) 66
Register 2: Cortex General-Purpose Register 1 (R1) 66
Register 3: Cortex General-Purpose Register 2 (R2) 66
Register 4: Cortex General-Purpose Register 3 (R3) 66
Register 5: Cortex General-Purpose Register 4 (R4) 66
Register 6: Cortex General-Purpose Register 5 (R5) 66
Register 7: Cortex General-Purpose Register 6 (R6) 66
Register 8: Cortex General-Purpose Register 7 (R7) 66
Register 9: Cortex General-Purpose Register 8 (R8) 66
Register 10: Cortex General-Purpose Register 9 (R9) 66
Register 11: Cortex General-Purpose Register 10 (R10) 66
Register 12: Cortex General-Purpose Register 11 (R11) 66
Register 13: Cortex General-Purpose Register 12 (R12) 66
Register 14: Stack Pointer (SP) 67
Register 15: Link Register (LR) 68
Register 16: Program Counter (PC) 69
Register 17: Program Status Register (PSR) 70
Register 18: Priority Mask Register (PRIMASK) 74
Register 19: Fault Mask Register (FAULTMASK) 75
Register 20: Base Priority Mask Register (BASEPRI) 76
Register 21: Control Register (CONTROL) 77
Register 22: Floating-Point Status Control (FPSC) 79
Cortex-M4 Peripherals 111
Register 1: SysTick Control and Status Register (STCTRL), offset 0x010 127
Register 2: SysTick Reload Value Register (STRELOAD), offset 0x014 129
Register 3: SysTick Current Value Register (STCURRENT), offset 0x018 130
Register 4: Interrupt 0-31 Set Enable (EN0), offset 0x100 131
Register 5: Interrupt 32-63 Set Enable (EN1), offset 0x104 131
Register 6: Interrupt 64-95 Set Enable (EN2), offset 0x108 131
Register 7: Interrupt 96-127 Set Enable (EN3), offset 0x10C 131
Register 8: Interrupt 128-138 Set Enable (EN4), offset 0x110 132
Register 9: Interrupt 0-31 Clear Enable (DIS0), offset 0x180 133
Register 10: Interrupt 32-63 Clear Enable (DIS1), offset 0x184 133
Register 11: Interrupt 64-95 Clear Enable (DIS2), offset 0x188 133
Register 12: Interrupt 96-127 Clear Enable (DIS3), offset 0x18C 133
Register 13: Interrupt 128-138 Clear Enable (DIS4), offset 0x190 134
Register 14: Interrupt 0-31 Set Pending (PEND0), offset 0x200 135
Register 15: Interrupt 32-63 Set Pending (PEND1), offset 0x204 135
Register 16: Interrupt 64-95 Set Pending (PEND2), offset 0x208 135
Register 17: Interrupt 96-127 Set Pending (PEND3), offset 0x20C 135
Register 18: Interrupt 128-138 Set Pending (PEND4), offset 0x210 136
Register 19: Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280 137
Register 20: Interrupt 32-63 Clear Pending (UNPEND1), offset 0x284 137
Register 21: Interrupt 64-95 Clear Pending (UNPEND2), offset 0x288 137

Register 22: Interrupt 96-127 Clear Pending (UNPEND3), offset 0x28C 137
Register 23: Interrupt 128-138 Clear Pending (UNPEND4), offset 0x290 138
Register 24: Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300 139
Register 25: Interrupt 32-63 Active Bit (ACTIVE1), offset 0x304 139
Register 26: Interrupt 64-95 Active Bit (ACTIVE2), offset 0x308 139
Register 27: Interrupt 96-127 Active Bit (ACTIVE3), offset 0x30C 139
Register 28: Interrupt 128-138 Active Bit (ACTIVE4), offset 0x310 140
Register 29: Interrupt 0-3 Priority (PRI0), offset 0x400 141
Register 30: Interrupt 4-7 Priority (PRI1), offset 0x404 141
Register 31: Interrupt 8-11 Priority (PRI2), offset 0x408 141
Register 32: Interrupt 12-15 Priority (PRI3), offset 0x40C 141
Register 33: Interrupt 16-19 Priority (PRI4), offset 0x410 141
Register 34: Interrupt 20-23 Priority (PRI5), offset 0x414 141
Register 35: Interrupt 24-27 Priority (PRI6), offset 0x418 141
Register 36: Interrupt 28-31 Priority (PRI7), offset 0x41C 141
Register 37: Interrupt 32-35 Priority (PRI8), offset 0x420 141
Register 38: Interrupt 36-39 Priority (PRI9), offset 0x424 141
Register 39: Interrupt 40-43 Priority (PRI10), offset 0x428 141
Register 40: Interrupt 44-47 Priority (PRI11), offset 0x42C 141
Register 41: Interrupt 48-51 Priority (PRI12), offset 0x430 141
Register 42: Interrupt 52-55 Priority (PRI13), offset 0x434 141
Register 43: Interrupt 56-59 Priority (PRI14), offset 0x438 141
Register 44: Interrupt 60-63 Priority (PRI15), offset 0x43C 141
Register 45: Interrupt 64-67 Priority (PRI16), offset 0x440 143
Register 46: Interrupt 68-71 Priority (PRI17), offset 0x444 143
Register 47: Interrupt 72-75 Priority (PRI18), offset 0x448 143
Register 48: Interrupt 76-79 Priority (PRI19), offset 0x44C 143
Register 49: Interrupt 80-83 Priority (PRI20), offset 0x450 143
Register 50: Interrupt 84-87 Priority (PRI21), offset 0x454 143
Register 51: Interrupt 88-91 Priority (PRI22), offset 0x458 143
Register 52: Interrupt 92-95 Priority (PRI23), offset 0x45C 143
Register 53: Interrupt 96-99 Priority (PRI24), offset 0x460 143
Register 54: Interrupt 100-103 Priority (PRI25), offset 0x464 143
Register 55: Interrupt 104-107 Priority (PRI26), offset 0x468 143
Register 56: Interrupt 108-111 Priority (PRI27), offset 0x46C 143
Register 57: Interrupt 112-115 Priority (PRI28), offset 0x470 143
Register 58: Interrupt 116-119 Priority (PRI29), offset 0x474 143
Register 59: Interrupt 120-123 Priority (PRI30), offset 0x478 143
Register 60: Interrupt 124-127 Priority (PRI31), offset 0x47C 143
Register 61: Interrupt 128-131 Priority (PRI32), offset 0x480 143
Register 62: Interrupt 132-135 Priority (PRI33), offset 0x484 143
Register 63: Interrupt 136-138 Priority (PRI34), offset 0x488 143
Register 64: Software Trigger Interrupt (SWTRIG), offset 0xF00 145
Register 65: Auxiliary Control (ACTLR), offset 0x008 146
Register 66: CPU ID Base (CPUID), offset 0xD00 148
Register 67: Interrupt Control and State (INTCTRL), offset 0xD04 149
Register 68: Vector Table Offset (VTABLE), offset 0xD08 152
Register 69: Application Interrupt and Reset Control (APINT), offset 0xD0C 153

Register 70: System Control (SYSCTRL), offset 0xD10 155
Register 71: Configuration and Control (CFGCTRL), offset 0xD14 157
Register 72: System Handler Priority 1 (SYSPRI1), offset 0xD18 159
Register 73: System Handler Priority 2 (SYSPRI2), offset 0xD1C 160
Register 74: System Handler Priority 3 (SYSPRI3), offset 0xD20 161
Register 75: System Handler Control and State (SYSHNDCTRL), offset 0xD24 162
Register 76: Configurable Fault Status (FAULTSTAT), offset 0xD28 166
Register 77: Hard Fault Status (HFAULTSTAT), offset 0xD2C 172
Register 78: Memory Management Fault Address (MMADDR), offset 0xD34 173
Register 79: Bus Fault Address (FAULTADDR), offset 0xD38 174
Register 80: MPU Type (MPUTYPE), offset 0xD90 175
Register 81: MPU Control (MPUCTRL), offset 0xD94 176
Register 82: MPU Region Number (MPUNUMBER), offset 0xD98 178
Register 83: MPU Region Base Address (MPUBASE), offset 0xD9C 179
Register 84: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4 179
Register 85: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC 179
Register 86: MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4 179
Register 87: MPU Region Attribute and Size (MPUATTR), offset 0xDA0 181
Register 88: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8 181
Register 89: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0 181
Register 90: MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8 181
Register 91: Coprocessor Access Control (CPAC), offset 0xD88 184
Register 92: Floating-Point Context Control (FPCC), offset 0xF34 185
Register 93: Floating-Point Context Address (FPCA), offset 0xF38 187
Register 94: Floating-Point Default Status Control (FPDSC), offset 0xF3C 188

System Control 201
Register 1: Device Identification 0 (DID0), offset 0x000 225
Register 2: Device Identification 1 (DID1), offset 0x004 227
Register 3: Brown-Out Reset Control (PBORCTL), offset 0x030 230
Register 4: Raw Interrupt Status (RIS), offset 0x050 231
Register 5: Interrupt Mask Control (IMC), offset 0x054 234
Register 6: Masked Interrupt Status and Clear (MISC), offset 0x058 236
Register 7: Reset Cause (RESC), offset 0x05C 239
Register 8: Run-Mode Clock Configuration (RCC), offset 0x060 241
Register 9: GPIO High-Performance Bus Control (GPIOHBCTL), offset 0x06C 245
Register 10: Run-Mode Clock Configuration 2 (RCC2), offset 0x070 247
Register 11: Main Oscillator Control (MOSCCTL), offset 0x07C 250
Register 12: Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144 251
Register 13: System Properties (SYSPROP), offset 0x14C 253
Register 14: Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150 255
Register 15: PLL Frequency 0 (PLLFREQ0), offset 0x160 256
Register 16: PLL Frequency 1 (PLLFREQ1), offset 0x164 257
Register 17: PLL Status (PLLSTAT), offset 0x168 258
Register 18: Sleep Power Configuration (SLPPWRCFG), offset 0x188 259
Register 19: Deep-Sleep Power Configuration (DSLPPWRCFG), offset 0x18C 261
Register 20: LDO Sleep Power Control (LDOSPCTL), offset 0x1B4 263
Register 21: LDO Sleep Power Calibration (LDOSPCAL), offset 0x1B8 265
Register 22: LDO Deep-Sleep Power Control (LDODPCTL), offset 0x1BC 266

Register 23: LDO Deep-Sleep Power Calibration (LDODPCAL), offset 0x1C0 268
Register 24: Sleep / Deep-Sleep Power Mode Status (SDPMST), offset 0x1CC 269
Register 25: Watchdog Timer Peripheral Present (PPWD), offset 0x300 272
Register 26: 16/32-Bit General-Purpose Timer Peripheral Present (PPTIMER), offset 0x304 273
Register 27: General-Purpose Input/Output Peripheral Present (PPGPIO), offset 0x308 275
Register 28: Micro Direct Memory Access Peripheral Present (PPDMA), offset 0x30C 278
Register 29: Hibernation Peripheral Present (PPHIB), offset 0x314 279
Register 30: Universal Asynchronous Receiver/Transmitter Peripheral Present (PPUART), offset 0x318 280
Register 31: Synchronous Serial Interface Peripheral Present (PPSSI), offset 0x31C 282
Register 32: Inter-Integrated Circuit Peripheral Present (PPI2C), offset 0x320 284
Register 33: Universal Serial Bus Peripheral Present (PPUSB), offset 0x328 286
Register 34: Controller Area Network Peripheral Present (PPCAN), offset 0x334 287
Register 35: Analog-to-Digital Converter Peripheral Present (PPADC), offset 0x338 288
Register 36: Analog Comparator Peripheral Present (PPACMP), offset 0x33C 289
Register 37: Pulse Width Modulator Peripheral Present (PPPWM), offset 0x340 290
Register 38: Quadrature Encoder Interface Peripheral Present (PPQEI), offset 0x344 291
Register 39: EEPROM Peripheral Present (PPEEPROM), offset 0x358 292
Register 40: 32/64-Bit Wide General-Purpose Timer Peripheral Present (PPWTIMER), offset 0x35C 293
Register 41: Watchdog Timer Software Reset (SRWD), offset 0x500 295
Register 42: 16/32-Bit General-Purpose Timer Software Reset (SRTIMER), offset 0x504 297
Register 43: General-Purpose Input/Output Software Reset (SRGPIO), offset 0x508 299
Register 44: Micro Direct Memory Access Software Reset (SRDMA), offset 0x50C 301
Register 45: Universal Asynchronous Receiver/Transmitter Software Reset (SRUART), offset 0x518 302
Register 46: Synchronous Serial Interface Software Reset (SRSSI), offset 0x51C 304
Register 47: Inter-Integrated Circuit Software Reset (SRI2C), offset 0x520 306
Register 48: Universal Serial Bus Software Reset (SRUSB), offset 0x528 308
Register 49: Controller Area Network Software Reset (SRCAN), offset 0x534 309
Register 50: Analog-to-Digital Converter Software Reset (SRADC), offset 0x538 310
Register 51: Analog Comparator Software Reset (SRACMP), offset 0x53C 312
Register 52: EEPROM Software Reset (SREEPROM), offset 0x558 313
Register 53: 32/64-Bit Wide General-Purpose Timer Software Reset (SRWTIMER), offset 0x55C 314
Register 54: Watchdog Timer Run Mode Clock Gating Control (RCGCWD), offset 0x600 316
Register 55: 16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER), offset 0x604 317
Register 56: General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO), offset 0x608 319
Register 57: Micro Direct Memory Access Run Mode Clock Gating Control (RCGCDMA), offset 0x60C 321
Register 58: Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control (RCGCUART), offset 0x618 322
Register 59: Synchronous Serial Interface Run Mode Clock Gating Control (RCGCSSI), offset 0x61C 324
Register 60: Inter-Integrated Circuit Run Mode Clock Gating Control (RCGCI2C), offset 0x620 326
Register 61: Universal Serial Bus Run Mode Clock Gating Control (RCGCUSB), offset 0x628 328
Register 62: Controller Area Network Run Mode Clock Gating Control (RCGCCAN), offset 0x634 329
Register 63: Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC), offset 0x638 330
Register 64: Analog Comparator Run Mode Clock Gating Control (RCGCACMP), offset 0x63C 331
Register 65: EEPROM Run Mode Clock Gating Control (RCGCEEPROM), offset 0x658 332

Register 66: 32/64-Bit Wide General-Purpose Timer Run Mode Clock Gating Control (RCGCWTIMER), offset 0x65C 333
Register 67: Watchdog Timer Sleep Mode Clock Gating Control (SCGCWD), offset 0x700 335
Register 68: 16/32-Bit General-Purpose Timer Sleep Mode Clock Gating Control (SCGCTIMER), offset 0x704 336
Register 69: General-Purpose Input/Output Sleep Mode Clock Gating Control (SCGCGPIO), offset 0x708 338
Register 70: Micro Direct Memory Access Sleep Mode Clock Gating Control (SCGCDMA), offset 0x70C 340
Register 71: Universal Asynchronous Receiver/Transmitter Sleep Mode Clock Gating Control (SCGCUART), offset 0x718 341
Register 72: Synchronous Serial Interface Sleep Mode Clock Gating Control (SCGCSSI), offset 0x71C 343
Register 73: Inter-Integrated Circuit Sleep Mode Clock Gating Control (SCGCI2C), offset 0x720 345
Register 74: Universal Serial Bus Sleep Mode Clock Gating Control (SCGCUSB), offset 0x728 347
Register 75: Controller Area Network Sleep Mode Clock Gating Control (SCGCCAN), offset 0x734 348
Register 76: Analog-to-Digital Converter Sleep Mode Clock Gating Control (SCGCADC), offset 0x738 349
Register 77: Analog Comparator Sleep Mode Clock Gating Control (SCGCACMP), offset 0x73C 350
Register 78: EEPROM Sleep Mode Clock Gating Control (SCGCEEPROM), offset 0x758 351
Register 79: 32/64-Bit Wide General-Purpose Timer Sleep Mode Clock Gating Control (SCGCWTIMER), offset 0x75C 352
Register 80: Watchdog Timer Deep-Sleep Mode Clock Gating Control (DCGCWD), offset 0x800 354
Register 81: 16/32-Bit General-Purpose Timer Deep-Sleep Mode Clock Gating Control (DCGCTIMER), offset 0x804 355
Register 82: General-Purpose Input/Output Deep-Sleep Mode Clock Gating Control (DCGCGPIO), offset 0x808 357
Register 83: Micro Direct Memory Access Deep-Sleep Mode Clock Gating Control (DCGCDMA), offset 0x80C 359
Register 84: Universal Asynchronous Receiver/Transmitter Deep-Sleep Mode Clock Gating Control (DCGCUART), offset 0x818 360
Register 85: Synchronous Serial Interface Deep-Sleep Mode Clock Gating Control (DCGCSSI), offset 0x81C 362
Register 86: Inter-Integrated Circuit Deep-Sleep Mode Clock Gating Control (DCGCI2C), offset 0x820 364
Register 87: Universal Serial Bus Deep-Sleep Mode Clock Gating Control (DCGCUSB), offset 0x828 366
Register 88: Controller Area Network Deep-Sleep Mode Clock Gating Control (DCGCCAN), offset 0x834 367
Register 89: Analog-to-Digital Converter Deep-Sleep Mode Clock Gating Control (DCGCADC), offset 0x838 368
Register 90: Analog Comparator Deep-Sleep Mode Clock Gating Control (DCGCACMP), offset 0x83C 369
Register 91: EEPROM Deep-Sleep Mode Clock Gating Control (DCGCEEPROM), offset 0x858 370
Register 92: 32/64-Bit Wide General-Purpose Timer Deep-Sleep Mode Clock Gating Control (DCGCWTIMER), offset 0x85C 371
Register 93: Watchdog Timer Peripheral Ready (PRWD), offset 0xA00 373
Register 94: 16/32-Bit General-Purpose Timer Peripheral Ready (PRTIMER), offset 0xA04 374
Register 95: General-Purpose Input/Output Peripheral Ready (PRGPIO), offset 0xA08 376
Register 96: Micro Direct Memory Access Peripheral Ready (PRDMA), offset 0xA0C 378

Register 97: Universal Asynchronous Receiver/Transmitter Peripheral Ready (PRUART), offset 0xA18 379
Register 98: Synchronous Serial Interface Peripheral Ready (PRSSI), offset 0xA1C 381
Register 99: Inter-Integrated Circuit Peripheral Ready (PRI2C), offset 0xA20 383
Register 100: Universal Serial Bus Peripheral Ready (PRUSB), offset 0xA28 385
Register 101: Controller Area Network Peripheral Ready (PRCAN), offset 0xA34 386
Register 102: Analog-to-Digital Converter Peripheral Ready (PRADC), offset 0xA38 387
Register 103: Analog Comparator Peripheral Ready (PRACMP), offset 0xA3C 388
Register 104: EEPROM Peripheral Ready (PREEPROM), offset 0xA58 389
Register 105: 32/64-Bit Wide General-Purpose Timer Peripheral Ready (PRWTIMER), offset 0xA5C 390
Register 106: Device Capabilities 0 (DC0), offset 0x008 392
Register 107: Device Capabilities 1 (DC1), offset 0x010 394
Register 108: Device Capabilities 2 (DC2), offset 0x014 397
Register 109: Device Capabilities 3 (DC3), offset 0x018 400
Register 110: Device Capabilities 4 (DC4), offset 0x01C 404
Register 111: Device Capabilities 5 (DC5), offset 0x020 407
Register 112: Device Capabilities 6 (DC6), offset 0x024 409
Register 113: Device Capabilities 7 (DC7), offset 0x028 410
Register 114: Device Capabilities 8 (DC8), offset 0x02C 413
Register 115: Software Reset Control 0 (SRCR0), offset 0x040 416
Register 116: Software Reset Control 1 (SRCR1), offset 0x044 418
Register 117: Software Reset Control 2 (SRCR2), offset 0x048 421
Register 118: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100 423
Register 119: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104 426
Register 120: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108 429
Register 121: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110 432
Register 122: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114 434
Register 123: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118 437
Register 124: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120 439
Register 125: Deep-Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124 441
Register 126: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128 444
Register 127: Device Capabilities 9 (DC9), offset 0x190 446
Register 128: Non-Volatile Memory Information (NVMSTAT), offset 0x1A0 448

System Exception Module 449

Register 1: System Exception Raw Interrupt Status (SYSEXCRIS), offset 0x000 450
Register 2: System Exception Interrupt Mask (SYSEXCIM), offset 0x004 452
Register 3: System Exception Masked Interrupt Status (SYSEXCMIS), offset 0x008 454
Register 4: System Exception Interrupt Clear (SYSEXCIC), offset 0x00C 456

Internal Memory 457

Register 1: Flash Memory Address (FMA), offset 0x000 475
Register 2: Flash Memory Data (FMD), offset 0x004 476
Register 3: Flash Memory Control (FMC), offset 0x008 477
Register 4: Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C 479
Register 5: Flash Controller Interrupt Mask (FCIM), offset 0x010 482
Register 6: Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014 484
Register 7: Flash Memory Control 2 (FMC2), offset 0x020 487
Register 8: Flash Write Buffer Valid (FWBVAL), offset 0x030 488
Register 9: Flash Write Buffer n (FWBn), offset 0x100 - 0x17C 489

Register 10: Flash Size (FSIZE), offset 0xFC0 490
Register 11: SRAM Size (SSIZE), offset 0xFC4 491
Register 12: ROM Software Map (ROMSWMAP), offset 0xFCC 492
Register 13: EEPROM Size Information (EESIZE), offset 0x000 493
Register 14: EEPROM Current Block (EEBLOCK), offset 0x004 494
Register 15: EEPROM Current Offset (EEOFFSET), offset 0x008 495
Register 16: EEPROM Read-Write (EERDWR), offset 0x010 496
Register 17: EEPROM Read-Write with Increment (EERDWRINC), offset 0x014 497
Register 18: EEPROM Done Status (EEDONE), offset 0x018 498
Register 19: EEPROM Support Control and Status (EESUPP), offset 0x01C 500
Register 20: EEPROM Unlock (EEUNLOCK), offset 0x020 502
Register 21: EEPROM Protection (EEPROT), offset 0x030 503
Register 22: EEPROM Password (EEPASS0), offset 0x034 505
Register 23: EEPROM Password (EEPASS1), offset 0x038 505
Register 24: EEPROM Password (EEPASS2), offset 0x03C 505
Register 25: EEPROM Interrupt (EEINT), offset 0x040 506
Register 26: EEPROM Block Hide (EEHIDE), offset 0x050 507
Register 27: EEPROM Debug Mass Erase (EEDBGME), offset 0x080 508
Register 28: EEPROM Peripheral Properties (EEPROMPP), offset 0xFC0 509
Register 29: ROM Control (RMCTL), offset 0x0F0 510
Register 30: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200 511
Register 31: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400 512
Register 32: Boot Configuration (BOOTCFG), offset 0x1D0 513
Register 33: User Register 0 (USER_REG0), offset 0x1E0 516
Register 34: User Register 1 (USER_REG1), offset 0x1E4 516
Register 35: User Register 2 (USER_REG2), offset 0x1E8 516
Register 36: User Register 3 (USER_REG3), offset 0x1EC 516

Micro Direct Memory Access (μDMA) 517

Register 1: DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000 541
Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004 542
Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008 543
Register 4: DMA Status (DMASTAT), offset 0x000 548
Register 5: DMA Configuration (DMACFG), offset 0x004 550
Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008 551
Register 7: DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C 552
Register 8: DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010 553
Register 9: DMA Channel Software Request (DMASWREQ), offset 0x014 554
Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018 555
Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C 556
Register 12: DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020 557
Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024 558
Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028 559
Register 15: DMA Channel Enable Clear (DMAENACLR), offset 0x02C 560
Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030 561
Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034 562
Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038 563
Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C 564
Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C 565

Register 21: DMA Channel Assignment (DMACHASGN), offset 0x500 566
Register 22: DMA Channel Interrupt Status (DMACHIS), offset 0x504 567
Register 23: DMA Channel Map Select 0 (DMACHMAP0), offset 0x510 568
Register 24: DMA Channel Map Select 1 (DMACHMAP1), offset 0x514 569
Register 25: DMA Channel Map Select 2 (DMACHMAP2), offset 0x518 570
Register 26: DMA Channel Map Select 3 (DMACHMAP3), offset 0x51C 571
Register 27: DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0 572
Register 28: DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4 573
Register 29: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8 574
Register 30: DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC 575
Register 31: DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0 576
Register 32: DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0 577
Register 33: DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4 578
Register 34: DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8 579
Register 35: DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC 580

General-Purpose Input/Outputs (GPIOs) 581

Register 1: GPIO Data (GPIODATA), offset 0x000 594
Register 2: GPIO Direction (GPIODIR), offset 0x400 595
Register 3: GPIO Interrupt Sense (GPIOIS), offset 0x404 596
Register 4: GPIO Interrupt Both Edges (GPIOIBE), offset 0x408 597
Register 5: GPIO Interrupt Event (GPIOIEV), offset 0x40C 598
Register 6: GPIO Interrupt Mask (GPIOIM), offset 0x410 599
Register 7: GPIO Raw Interrupt Status (GPIORIS), offset 0x414 600
Register 8: GPIO Masked Interrupt Status (GPIOMIS), offset 0x418 601
Register 9: GPIO Interrupt Clear (GPIOICR), offset 0x41C 602
Register 10: GPIO Alternate Function Select (GPIOAFSEL), offset 0x420 603
Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500 605
Register 12: GPIO 4-mA Drive Select (GPIODR4R), offset 0x504 606
Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508 607
Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C 608
Register 15: GPIO Pull-Up Select (GPIOPUR), offset 0x510 609
Register 16: GPIO Pull-Down Select (GPIOPDR), offset 0x514 611
Register 17: GPIO Slew Rate Control Select (GPIOSLR), offset 0x518 613
Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C 614
Register 19: GPIO Lock (GPIOLOCK), offset 0x520 616
Register 20: GPIO Commit (GPIOCR), offset 0x524 617
Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528 619
Register 22: GPIO Port Control (GPIOPCTL), offset 0x52C 620
Register 23: GPIO ADC Control (GPIOADCCTL), offset 0x530 622
Register 24: GPIO DMA Control (GPIODMACTL), offset 0x534 623
Register 25: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0 624
Register 26: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4 625
Register 27: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8 626
Register 28: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC 627
Register 29: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0 628
Register 30: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4 629
Register 31: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8 630
Register 32: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC 631

Register 33: GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0 632
Register 34: GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4 633
Register 35: GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8 634
Register 36: GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC 635
General-Purpose Timers 636
Register 1: GPTM Configuration (GPTMCFG), offset 0x000 659
Register 2: GPTM Timer A Mode (GPTMTAMR), offset 0x004 661
Register 3: GPTM Timer B Mode (GPTMTBMR), offset 0x008 665
Register 4: GPTM Control (GPTMCTL), offset 0x00C 669
Register 5: GPTM Synchronize (GPTMSYNC), offset 0x010 673
Register 6: GPTM Interrupt Mask (GPTMIMR), offset 0x018 677
Register 7: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C 680
Register 8: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020 683
Register 9: GPTM Interrupt Clear (GPTMICR), offset 0x024 686
Register 10: GPTM Timer A Interval Load (GPTMTAILR), offset 0x028 688
Register 11: GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C 689
Register 12: GPTM Timer A Match (GPTMTAMATCHR), offset 0x030 690
Register 13: GPTM Timer B Match (GPTMTBMATCHR), offset 0x034 691
Register 14: GPTM Timer A Prescale (GPTMTAPR), offset 0x038 692
Register 15: GPTM Timer B Prescale (GPTMTBPR), offset 0x03C 693
Register 16: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040 694
Register 17: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044 695
Register 18: GPTM Timer A (GPTMTAR), offset 0x048 696
Register 19: GPTM Timer B (GPTMTBR), offset 0x04C 697
Register 20: GPTM Timer A Value (GPTMTAV), offset 0x050 698
Register 21: GPTM Timer B Value (GPTMTBV), offset 0x054 699
Register 22: GPTM RTC Predivide (GPTMRTCPD), offset 0x058 700
Register 23: GPTM Timer A Prescale Snapshot (GPTMTAPS), offset 0x05C 701
Register 24: GPTM Timer B Prescale Snapshot (GPTMTBPS), offset 0x060 702
Register 25: GPTM Timer A Prescale Value (GPTMTAPV), offset 0x064 703
Register 26: GPTM Timer B Prescale Value (GPTMTBPV), offset 0x068 704
Register 27: GPTM Peripheral Properties (GPTMPP), offset 0xFC0 705
Watchdog Timers 706
Register 1: Watchdog Load (WDTLOAD), offset 0x000 710
Register 2: Watchdog Value (WDTVALUE), offset 0x004 711
Register 3: Watchdog Control (WDTCTL), offset 0x008 712
Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C 714
Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010 715
Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014 716
Register 7: Watchdog Test (WDTTEST), offset 0x418 717
Register 8: Watchdog Lock (WDTLOCK), offset 0xC00 718
Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0 719
Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4 720
Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8 721
Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC 722
Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0 723
Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4 724
Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8 725

Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC 726
Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0 727
Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4 728
Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8 729
Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC 730

Analog-to-Digital Converter (ADC) 731

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000 753
Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004 755
Register 3: ADC Interrupt Mask (ADCIM), offset 0x008 757
Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C 760
Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010 763
Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014 765
Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018 770
Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020 771
Register 9: ADC Sample Phase Control (ADCSPC), offset 0x024 773
Register 10: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028 775
Register 11: ADC Sample Averaging Control (ADCSAC), offset 0x030 777
Register 12: ADC Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034 778
Register 13: ADC Control (ADCCTL), offset 0x038 780
Register 14: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040 781
Register 15: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044 783
Register 16: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048 790
Register 17: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068 790
Register 18: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088 790
Register 19: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8 790
Register 20: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C 791
Register 21: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C 791
Register 22: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C 791
Register 23: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC 791
Register 24: ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050 793
Register 25: ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054 795
Register 26: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060 797
Register 27: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080 797
Register 28: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064 798
Register 29: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084 798
Register 30: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070 802
Register 31: ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090 802
Register 32: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074 803
Register 33: ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094 803
Register 34: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0 805
Register 35: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4 806
Register 36: ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0 808
Register 37: ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4 809
Register 38: ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00 810
Register 39: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00 815
Register 40: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04 815
Register 41: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08 815
Register 42: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C 815

Register 43: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10 815
Register 44: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14 815
Register 45: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18 815
Register 46: ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C 815
Register 47: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40 817
Register 48: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44 817
Register 49: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48 817
Register 50: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C 817
Register 51: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50 817
Register 52: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54 817
Register 53: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58 817
Register 54: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C 817
Register 55: ADC Peripheral Properties (ADCPP), offset 0xFC0 818
Register 56: ADC Peripheral Configuration (ADCPC), offset 0xFC4 820
Register 57: ADC Clock Configuration (ADCCC), offset 0xFC8 821

Universal Asynchronous Receivers/Transmitters (UARTs) 822

Register 1: UART Data (UARTDR), offset 0x000 835
Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004 837
Register 3: UART Flag (UARTFR), offset 0x018 840
Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020 842
Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024 843
Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028 844
Register 7: UART Line Control (UARTLCRH), offset 0x02C 845
Register 8: UART Control (UARTCTL), offset 0x030 847
Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034 851
Register 10: UART Interrupt Mask (UARTIM), offset 0x038 853
Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C 856
Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040 859
Register 13: UART Interrupt Clear (UARTICR), offset 0x044 862
Register 14: UART DMA Control (UARTDMACTL), offset 0x048 864
Register 15: UART 9-Bit Self Address (UART9BITADDR), offset 0x0A4 865
Register 16: UART 9-Bit Self Address Mask (UART9BITAMASK), offset 0x0A8 866
Register 17: UART Peripheral Properties (UARTPP), offset 0xFC0 867
Register 18: UART Clock Configuration (UARTCC), offset 0xFC8 868
Register 19: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0 869
Register 20: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4 870
Register 21: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8 871
Register 22: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC 872
Register 23: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0 873
Register 24: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4 874
Register 25: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8 875
Register 26: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC 876
Register 27: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0 877
Register 28: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4 878
Register 29: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8 879
Register 30: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC 880

Synchronous Serial Interface (SSI) 881

Register 1: SSI Control 0 (SSICR0), offset 0x000 898

Register 2: SSI Control 1 (SSICR1), offset 0x004 900
Register 3: SSI Data (SSIDR), offset 0x008 902
Register 4: SSI Status (SSISR), offset 0x00C 903
Register 5: SSI Clock Prescale (SSICPSR), offset 0x010 905
Register 6: SSI Interrupt Mask (SSIIM), offset 0x014 906
Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018 907
Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C 909
Register 9: SSI Interrupt Clear (SSIICR), offset 0x020 911
Register 10: SSI DMA Control (SSIDMACTL), offset 0x024 912
Register 11: SSI Clock Configuration (SSICC), offset 0xFC8 913
Register 12: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0 914
Register 13: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4 915
Register 14: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8 916
Register 15: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC 917
Register 16: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0 918
Register 17: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4 919
Register 18: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8 920
Register 19: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC 921
Register 20: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0 922
Register 21: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4 923
Register 22: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8 924
Register 23: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC 925

Inter-Integrated Circuit (I^{2}C) Interface 926

Register 1: I^{2}C Master Slave Address (I2CMSA), offset 0x000 948
Register 2: I^{2}C Master Control/Status (I2CMCS), offset 0x004 949
Register 3: I^{2}C Master Data (I2CMDR), offset 0x008 954
Register 4: I^{2}C Master Timer Period (I2CMTPR), offset 0x00C 955
Register 5: I^{2}C Master Interrupt Mask (I2CMIMR), offset 0x010 956
Register 6: I^{2}C Master Raw Interrupt Status (I2CMRIS), offset 0x014 957
Register 7: I^{2}C Master Masked Interrupt Status (I2CMMIS), offset 0x018 958
Register 8: I^{2}C Master Interrupt Clear (I2CMICR), offset 0x01C 959
Register 9: I^{2}C Master Configuration (I2CMCR), offset 0x020 960
Register 10: I^{2}C Master Clock Low Timeout Count (I2CMCLKOCNT), offset 0x024 962
Register 11: I^{2}C Master Bus Monitor (I2CMBMON), offset 0x02C 963
Register 12: I^{2}C Master Configuration 2 (I2CMCR2), offset 0x038 964
Register 13: I^{2}C Slave Own Address (I2CSOAR), offset 0x800 965
Register 14: I^{2}C Slave Control/Status (I2CSCSR), offset 0x804 966
Register 15: I^{2}C Slave Data (I2CSDR), offset 0x808 968
Register 16: I^{2}C Slave Interrupt Mask (I2CSIMR), offset 0x80C 969
Register 17: I^{2}C Slave Raw Interrupt Status (I2CSRIS), offset 0x810 970
Register 18: I^{2}C Slave Masked Interrupt Status (I2CSMIS), offset 0x814 971
Register 19: I^{2}C Slave Interrupt Clear (I2CSICR), offset 0x818 972
Register 20: I^{2}C Slave Own Address 2 (I2CSOAR2), offset 0x81C 973
Register 21: I^{2}C Slave ACK Control (I2CSACKCTL), offset 0x820 974
Register 22: I^{2}C Peripheral Properties (I2CPP), offset 0xFC0 975
Register 23: I^{2}C Peripheral Configuration (I2CPC), offset 0xFC4 976

Controller Area Network (CAN) Module 977

Register 1: CAN Control (CANCTL), offset 0x000 998
Register 2: CAN Status (CANSTS), offset 0x004 1000
Register 3: CAN Error Counter (CANERR), offset 0x008 1003
Register 4: CAN Bit Timing (CANBIT), offset 0x00C 1004
Register 5: CAN Interrupt (CANINT), offset 0x010 1005
Register 6: CAN Test (CANTST), offset 0x014 1006
Register 7: CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018 1008
Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020 1009
Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080 1009
Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024 1010
Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084 1010
Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028 1013
Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088 1013
Register 14: CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C 1014
Register 15: CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C 1014
Register 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030 1016
Register 17: CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090 1016
Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034 1017
Register 19: CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094 1017
Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038 1019
Register 21: CAN IF2 Message Control (CANIF2MCTL), offset 0x098 1019
Register 22: CAN IF1 Data A1 (CANIF1DA1), offset 0x03C 1022
Register 23: CAN IF1 Data A2 (CANIF1DA2), offset 0x040 1022
Register 24: CAN IF1 Data B1 (CANIF1DB1), offset 0x044 1022
Register 25: CAN IF1 Data B2 (CANIF1DB2), offset 0x048 1022
Register 26: CAN IF2 Data A1 (CANIF2DA1), offset 0x09C 1022
Register 27: CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0 1022
Register 28: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4 1022
Register 29: CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8 1022
Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100 1023
Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104 1023
Register 32: CAN New Data 1 (CANNWDA1), offset 0x120 1024
Register 33: CAN New Data 2 (CANNWDA2), offset 0x124 1024
Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140 1025
Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144 1025
Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160 1026
Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164 1026

Universal Serial Bus (USB) Controller 1027

Register 1: USB Device Functional Address (USBFADDR), offset 0x000 1039
Register 2: USB Power (USBPOWER), offset 0x001 1040
Register 3: USB Transmit Interrupt Status (USBTXIS), offset 0x002 1042
Register 4: USB Receive Interrupt Status (USBRXIS), offset 0x004 1044
Register 5: USB Transmit Interrupt Enable (USBTXIE), offset 0x006 1045
Register 6: USB Receive Interrupt Enable (USBRXIE), offset 0x008 1047
Register 7: USB General Interrupt Status (USBIS), offset 0x00A 1048
Register 8: USB Interrupt Enable (USBIE), offset 0x00B 1049
Register 9: USB Frame Value (USBFRAME), offset 0x00C 1050

Register 10: USB Endpoint Index (USBEPIDX), offset 0x00E 1051
Register 11: USB Test Mode (USBTEST), offset 0x00F 1052
Register 12: USB FIFO Endpoint 0 (USBFIFO0), offset 0x020 1053
Register 13: USB FIFO Endpoint 1 (USBFIFO1), offset 0x024 1053
Register 14: USB FIFO Endpoint 2 (USBFIFO2), offset 0x028 1053
Register 15: USB FIFO Endpoint 3 (USBFIFO3), offset 0x02C 1053
Register 16: USB FIFO Endpoint 4 (USBFIFO4), offset 0x030 1053
Register 17: USB FIFO Endpoint 5 (USBFIFO5), offset 0x034 1053
Register 18: USB FIFO Endpoint 6 (USBFIFO6), offset 0x038 1053
Register 19: USB FIFO Endpoint 7 (USBFIFO7), offset 0x03C 1053
Register 20: USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062 1054
Register 21: USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063 1054
Register 22: USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064 1055
Register 23: USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066 1055
Register 24: USB Connect Timing (USBCONTIM), offset 0x07A 1056
Register 25: USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF), offset 0x07D 1057
Register 26: USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF), offset 0x07E 1058
Register 27: USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110 1059
Register 28: USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120 1059
Register 29: USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130 1059
Register 30: USB Maximum Transmit Data Endpoint 4 (USBTXMAXP4), offset 0x140 1059
Register 31: USB Maximum Transmit Data Endpoint 5 (USBTXMAXP5), offset 0x150 1059
Register 32: USB Maximum Transmit Data Endpoint 6 (USBTXMAXP6), offset 0x160 1059
Register 33: USB Maximum Transmit Data Endpoint 7 (USBTXMAXP7), offset 0x170 1059
Register 34: USB Control and Status Endpoint 0 Low (USBCSRL0), offset 0x102 1060
Register 35: USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103 1062
Register 36: USB Receive Byte Count Endpoint 0 (USBCOUNT0), offset 0x108 1063
Register 37: USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1), offset 0x112 1064
Register 38: USB Transmit Control and Status Endpoint 2 Low (USBTXCSRL2), offset 0x122 1064
Register 39: USB Transmit Control and Status Endpoint 3 Low (USBTXCSRL3), offset 0x132 1064
Register 40: USB Transmit Control and Status Endpoint 4 Low (USBTXCSRL4), offset 0x142 1064
Register 41: USB Transmit Control and Status Endpoint 5 Low (USBTXCSRL5), offset 0x152 1064
Register 42: USB Transmit Control and Status Endpoint 6 Low (USBTXCSRL6), offset 0x162 1064
Register 43: USB Transmit Control and Status Endpoint 7 Low (USBTXCSRL7), offset 0x172 1064
Register 44: USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1), offset 0x113 1066
Register 45: USB Transmit Control and Status Endpoint 2 High (USBTXCSRH2), offset 0x123 1066
Register 46: USB Transmit Control and Status Endpoint 3 High (USBTXCSRH3), offset 0x133 1066
Register 47: USB Transmit Control and Status Endpoint 4 High (USBTXCSRH4), offset 0x143 1066
Register 48: USB Transmit Control and Status Endpoint 5 High (USBTXCSRH5), offset 0x153 1066
Register 49: USB Transmit Control and Status Endpoint 6 High (USBTXCSRH6), offset 0x163 1066
Register 50: USB Transmit Control and Status Endpoint 7 High (USBTXCSRH7), offset 0x173 1066
Register 51: USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114 1068
Register 52: USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124 1068
Register 53: USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134 1068
Register 54: USB Maximum Receive Data Endpoint 4 (USBRXMAXP4), offset 0x144 1068
Register 55: USB Maximum Receive Data Endpoint 5 (USBRXMAXP5), offset 0x154 1068
Register 56: USB Maximum Receive Data Endpoint 6 (USBRXMAXP6), offset 0x164 1068
Register 57: USB Maximum Receive Data Endpoint 7 (USBRXMAXP7), offset 0x174 1068

Register 58: USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1), offset 0x116 1069
Register 59: USB Receive Control and Status Endpoint 2 Low (USBRXCSRL2), offset 0x126 1069
Register 60: USB Receive Control and Status Endpoint 3 Low (USBRXCSRL3), offset 0x136 1069
Register 61: USB Receive Control and Status Endpoint 4 Low (USBRXCSRL4), offset 0x146 1069
Register 62: USB Receive Control and Status Endpoint 5 Low (USBRXCSRL5), offset 0x156 1069
Register 63: USB Receive Control and Status Endpoint 6 Low (USBRXCSRL6), offset 0x166 1069
Register 64: USB Receive Control and Status Endpoint 7 Low (USBRXCSRL7), offset 0x176 1069
Register 65: USB Receive Control and Status Endpoint 1 High (USBRXCSRH1), offset 0x117 1072
Register 66: USB Receive Control and Status Endpoint 2 High (USBRXCSRH2), offset 0x127 1072
Register 67: USB Receive Control and Status Endpoint 3 High (USBRXCSRH3), offset 0x137 1072
Register 68: USB Receive Control and Status Endpoint 4 High (USBRXCSRH4), offset 0x147 1072
Register 69: USB Receive Control and Status Endpoint 5 High (USBRXCSRH5), offset 0x157 1072
Register 70: USB Receive Control and Status Endpoint 6 High (USBRXCSRH6), offset 0x167 1072
Register 71: USB Receive Control and Status Endpoint 7 High (USBRXCSRH7), offset 0x177 1072
Register 72: USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118 1074
Register 73: USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128 1074
Register 74: USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138 1074
Register 75: USB Receive Byte Count Endpoint 4 (USBRXCOUNT4), offset 0x148 1074
Register 76: USB Receive Byte Count Endpoint 5 (USBRXCOUNT5), offset 0x158 1074
Register 77: USB Receive Byte Count Endpoint 6 (USBRXCOUNT6), offset 0x168 1074
Register 78: USB Receive Byte Count Endpoint 7 (USBRXCOUNT7), offset 0x178 1074
Register 79: USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS), offset 0x340 1075
Register 80: USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS), offset 0x342 1076
Register 81: USB Device RESUME Raw Interrupt Status (USBDRRIS), offset 0x410 1077
Register 82: USB Device RESUME Interrupt Mask (USBDRIM), offset 0x414 1078
Register 83: USB Device RESUME Interrupt Status and Clear (USBDRISC), offset 0x418 1079
Register 84: USB DMA Select (USBDMASEL), offset 0x450 1080
Register 85: USB Peripheral Properties (USBPP), offset 0xFC0 1082
Analog Comparators 1083
Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000 1090
Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004 1091
Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008 1092
Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010 1093
Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020 1094
Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040 1094
Register 7: Analog Comparator Control 0 (ACCTL0), offset 0x024 1095
Register 8: Analog Comparator Control 1 (ACCTL1), offset 0x044 1095
Register 9: Analog Comparator Peripheral Properties (ACMPPP), offset 0xFC0 1097

Revision History

The revision history table notes changes made between the indicated revisions of the TM4C1232C3PM data sheet.

Table 1. Revision History

Date	Revision	Description
June 2014	15842.2741	■ In System Control Chapter, corrected description for `MINSYSDIV` bitfield in **Device Capabilities 1 (DC1)** legacy register. ■ In Timers chapter, removed erroneous references to `TCACT` bit field. ■ In SSI chapter, corrected that during idle periods the transmit data line SSInTx is tristated. ■ In Package Information appendix: – Corrected Key to Part Numbers diagram. – Moved Orderable Part Numbers table to addendum. – Deleted Packaging Materials section and put into separate packaging document. ■ Additional minor data sheet clarifications and corrections.
March 2014	15741.2722	■ In the Internal Memory chapter, in the EEPROM section: – Added section on soft reset handling. – Added important information on EEPROM initialization and configuration. ■ In the DMA chapter, added information regarding interrupts and transfers from the UART or SSI modules. ■ In the GPIO chapter: – Corrected table GPIO Pins with Special Considerations. – Added information on preventing false interrupts. – Corrected **GPIOAMSEL** register to be 8 bits. ■ In the Timer chapter: – Clarified initialization and configuration for Input-Edge Count mode. – Clarified behavior of `TnMIE` and `TnCINTD` bits in the **GPTM Timer n Mode (GPTMTnMR)** register. ■ In the USB chapter, added note to SUSPEND section regarding bus-powered devices. ■ In the Electrical Characteristics chapter: – In table Reset Characteristics, clarified internal reset time parameter values. – In table Hibernation Oscillator Input Characteristics, added parameter C_{INSE} Input capacitance. – In tables Hibernation Oscillator Input Characteristics and Main Oscillator Input Characteristics, removed parameter C0 Crystal shunt capacitance. – Updated table Crystal Parameters. – In table GPIO Module Characteristics, added parameter C_{GPIO} GPIO Digital Input Capacitance. – Added table PWM Timing Characteristics. ■ In the Package Information appendix: – Updated Orderable Devices section to reflect silicon revision 7 part numbers. – Added Tape and Reel pin 1 location. ■ Additional minor data sheet clarifications and corrections.
November 2013	15553.2700	■ In System Control chapter, clarified PIOSC features and accuracy. ■ In Watchdog Timers chapter, clarified **Watchdog Control (WDTCTL)** register description. ■ In ADC chapter:

Table 1. Revision History *(continued)*

Date	Revision	Description
		– Clarified functionality when using an ADC digital comparator as a fault source. – Clarified signals used for ADC voltage reference. – Corrected `VREF` bit in **ADC Control (ADCCTL)** register from 2-bit field [1:0] to 1-bit field [0]. ■ In UART chapter, clarified DMA operation. ■ In SSI chapter: – Corrected timing guidelines in figures "Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0" and "Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0". – Clarified SSI Initialization and Configuration. – Corrected bit 3 in **SSI Control 1 (SSICR1)** register from `SOD` (SSI Slave Mode Output Disable) to reserved. ■ In Signal Tables chapter: – In Unused Signals table, corrected preferred and acceptable practices for `RST` pin. – Clarified `GNDX` pin description. ■ In Electrical Characteristics chapter: – In Power-On and Brown-Out Levels table, corrected T_{VDDC_RISE} parameter min and max values. – In PIOSC Clock Characteristics table, clarified F_{PIOSC} parameter values by defining values for both factory calibration and recalibration. Also added PIOSC startup time parameter to table. – In Main Oscillator Specifications section, corrected minimum value for External load capacitance on `OSC0`, `OSC1` pins. Also added two 25-MHz crystals to Crystal Parameters table. – Corrected figure "Master Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1". – In I^2C Characteristics table, clarified T_{DH} data hold time parameter values by defining values for both slave and master. In addition, added parameter I10 T_{DV} data valid. – Modified figure "I2C Timing" to add new parameter I10. ■ In Packaging Information appendix, added Packaging Materials figures.
July 16, 2013	15033.2672	■ In the Electrical Characteristics chapter: – Added maximum junction temperature to Maximum Ratings table. Also moved Unpowered storage temperature range parameter to this table. – In SSI Characteristics table, corrected values for T_{RXDMS}, T_{RXDMH}, and T_{RXDSSU}. Also clarified footnotes to table. – Corrected parameter numbers in figures "Master Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1" and "Slave Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1". ■ Additional minor data sheet clarifications and corrections.
July 2013	14995.2667	■ In the System Control chapter, corrected resets for bits [7:4] in **System Properties (SYSPROP)** register. ■ In the Internal Memory chapter, removed the `INVPL` bit from the **EEPROM Done Status (EEDONE)** register.

Table 1. Revision History *(continued)*

Date	Revision	Description
		■ In the uDMA chapter, in the µDMA Channel Assignments table, corrected names of timers 6-11 to wide timers 0-5. ■ In the Timers chapter: – Clarified that the timer must be configured for one-shot or periodic time-out mode to produce an ADC trigger assertion and that the GPTM does not generate triggers for match, compare events or compare match events. – Added a step in the RTC Mode initialization and configuration: If the timer has been operating in a different mode prior to this, clear any residual set bits in the **GPTM Timer n Mode (GPTMTnMR)** register before reconfiguring. ■ In the Watchdog Timer chapter, added a note that locking the watchdog registers using the **WDTLOCK** register does not affect the **WDTICR** register and allows interrupts to always be serviced. ■ In the SSI chapter, clarified note in Bit Rate Generation section to indicate that the System Clock or the PIOSC can be used as the source for `SSIClk`. Also corrected to indicate maximum SSIClk limit in SSI slave mode as well as the fact that SYSCLK has to be at least 12 times that of SSICLk. ■ In the Electrical Characteristics chapter: – Moved Maximum Ratings and ESD Absolute Maximum Ratings to the front of the chapter. – Added V_{BATRMP} parameter to Maximum Ratings and Hibernation Module Battery Characteristics tables. – Added ambient and junction temperatures to Temperature Characteristics table and clarified values in Thermal Characteristics table. – Added clarifying footnote to V_{VDD_POK} parameter in Power-On and Brown-Out Levels table. – In the Flash Memory and EEPROM Characteristics tables, added a parameter for page/mass erase times for 10k cycles and corrected existing values for all page and mass erase parameters. – Corrected DNL max value in ADC Electrical Characteristics table. – In the SSI Characteristics table, changed parameter names for S7-S14, provided a max number instead of a min for S7, and corrected values for S9-S14. – Replaced figure "SSI Timing for SPI Frame Format (FRF=00), with SPH=1" with two figures, one for Master Mode and one for Slave Mode. – Updated and added values to the table Table 21-37 on page 1161. ■ In the Package Information appendix, moved orderable devices table from addendum to appendix, clarified part markings and moved packaging diagram from addendum to appendix. ■ Additional minor data sheet clarifications and corrections.

About This Document

This data sheet provides reference information for the TM4C1232C3PM microcontroller, describing the functional blocks of the system-on-chip (SoC) device designed around the ARM® Cortex™-M4F core.

Audience

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

About This Manual

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

Related Documents

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

- *Tiva™ C Series TM4C123x Silicon Errata (literature number SPMZ849)*
- *TivaWare™ Boot Loader for C Series User's Guide (literature number SPMU301)*
- *TivaWare™ Graphics Library for C Series User's Guide (literature number SPMU300)*
- *TivaWare™ for C Series Release Notes (literature number SPMU299)*
- *TivaWare™ Peripheral Driver Library for C Series User's Guide (literature number SPMU298)*
- *TivaWare™ USB Library for C Series User's Guide (literature number SPMU297)*
- *Tiva™ C Series TM4C123x ROM User's Guide (literature number SPMU367)*

The following related documents may also be useful:

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

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

Documentation Conventions

This document uses the conventions shown in Table 2 on page 36.

Table 2. Documentation Conventions

Notation	Meaning
General Register Notation	
REGISTER	APB registers are indicated in uppercase bold. For example, **PBORCTL** is the Power-On and Brown-Out Reset Control register. If a register name contains a lowercase n, it represents more than one register. For example, **SRCRn** represents any (or all) of the three Software Reset Control registers: **SRCR0, SRCR1** , and **SRCR2**.
bit	A single bit in a register.
bit field	Two or more consecutive and related bits.
offset 0x*nnn*	A hexadecimal increment to a register's address, relative to that module's base address as specified in Table 2-4 on page 81.
Register *N*	Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software.
reserved	Register bits marked *reserved* are reserved for future use. In most cases, reserved bits are set to 0; however, user software should not rely on the value of a reserved bit. To provide software compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
yy:xx	The range of register bits inclusive from xx to yy. For example, 31:15 means bits 15 through 31 in that register.
Register Bit/Field Types	This value in the register bit diagram indicates whether software running on the controller can change the value of the bit field.
RC	Software can read this field. The bit or field is cleared by hardware after reading the bit/field.
RO	Software can read this field. Always write the chip reset value.
RW	Software can read or write this field.
RWC	Software can read or write this field. Writing to it with any value clears the register.
RW1C	Software can read or write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. This register type is primarily used for clearing interrupt status bits where the read operation provides the interrupt status and the write of the read value clears only the interrupts being reported at the time the register was read.
RW1S	Software can read or write a 1 to this field. A write of a 0 to a RW1S bit does not affect the bit value in the register.
W1C	Software can write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. A read of the register returns no meaningful data. This register is typically used to clear the corresponding bit in an interrupt register.
WO	Only a write by software is valid; a read of the register returns no meaningful data.
Register Bit/Field Reset Value	This value in the register bit diagram shows the bit/field value after any reset, unless noted.
0	Bit cleared to 0 on chip reset.
1	Bit set to 1 on chip reset.
-	Nondeterministic.
Pin/Signal Notation	
[ ]	Pin alternate function; a pin defaults to the signal without the brackets.
pin	Refers to the physical connection on the package.
signal	Refers to the electrical signal encoding of a pin.

Table 2. Documentation Conventions *(continued)*

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

1 Architectural Overview

Texas Instrument's Tiva™ C Series microcontrollers provide designers a high-performance ARM® Cortex™-M-based architecture with a broad set of integration capabilities and a strong ecosystem of software and development tools. Targeting performance and flexibility, the Tiva™ C Series architecture offers a 80 MHz Cortex-M with FPU, a variety of integrated memories and multiple programmable GPIO. Tiva™ C Series devices offer consumers compelling cost-effective solutions by integrating application-specific peripherals and providing a comprehensive library of software tools which minimize board costs and design-cycle time. Offering quicker time-to-market and cost savings, the Tiva™ C Series microcontrollers are the leading choice in high-performance 32-bit applications.

This chapter contains an overview of the Tiva™ C Series microcontrollers as well as details on the TM4C1232C3PM microcontroller:

- “Tiva™ C Series Overview” on page 38
- “TM4C1232C3PM Microcontroller Overview” on page 39
- “TM4C1232C3PM Microcontroller Features” on page 41
- “TM4C1232C3PM Microcontroller Hardware Details” on page 56
- “Kits” on page 57
- “Support Information” on page 57

1.1 Tiva™ C Series Overview

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

- Low power, hand-held smart devices
- Gaming equipment
- Home and commercial site monitoring and control
- Motion control
- Medical instrumentation
- Test and measurement equipment
- Factory automation
- Fire and security
- Smart Energy/Smart Grid solutions
- Intelligent lighting control
- Transportation

In addition, the TM4C1232C3PM 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, much of the TM4C1232C3PM microcontroller code is compatible to the Tiva™ C Series product line, providing flexibility across designs.

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

1.2 TM4C1232C3PM Microcontroller Overview

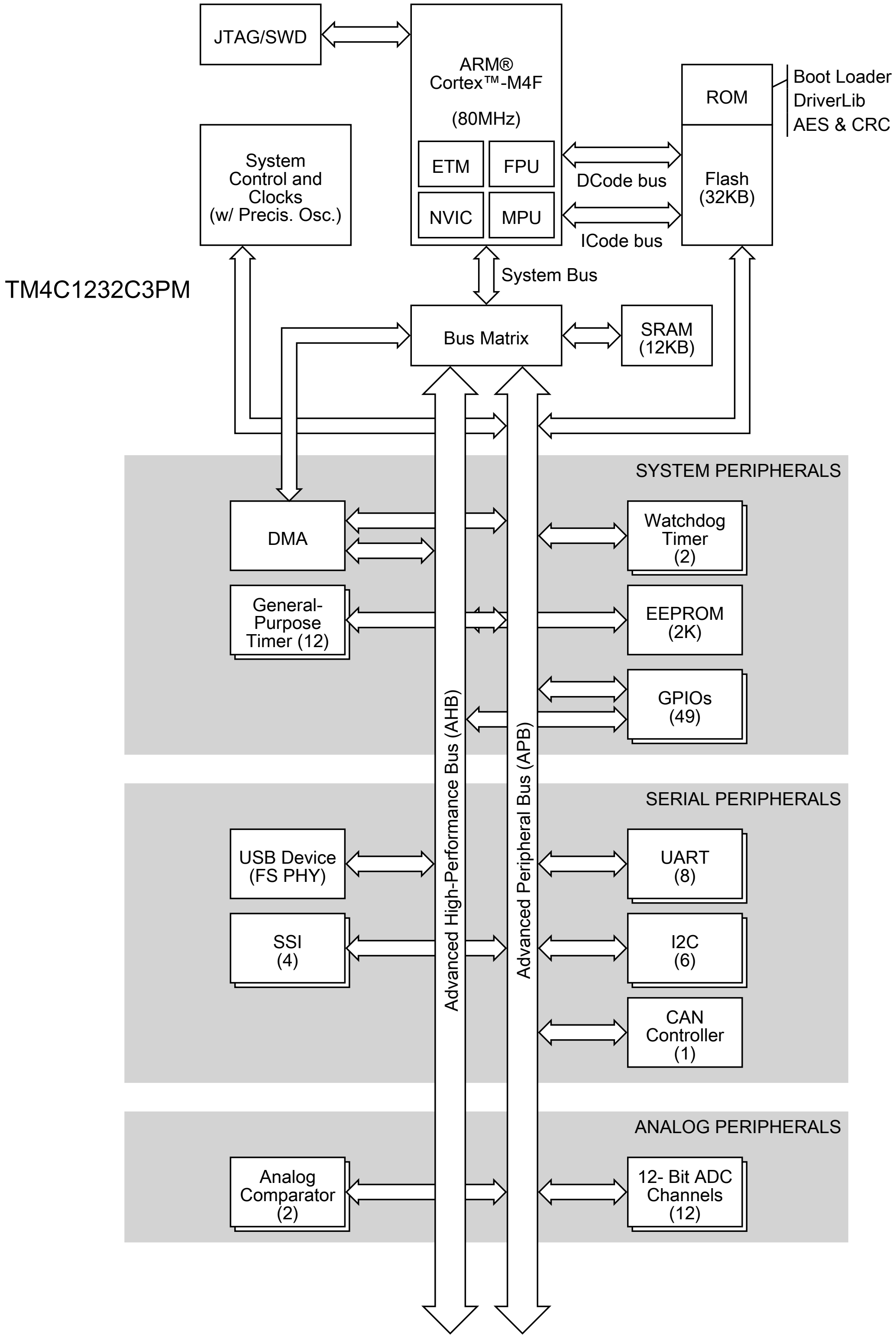
The TM4C1232C3PM microcontroller combines complex integration and high performance with the features shown in Table 1-1.

Table 1-1. TM4C1232C3PM Microcontroller Features

Feature	Description
Performance	
Core	ARM Cortex-M4F processor core
Performance	80-MHz operation; 100 DMIPS performance
Flash	32 KB single-cycle Flash memory
System SRAM	12 KB single-cycle SRAM
EEPROM	2KB of EEPROM
Internal ROM	Internal ROM loaded with TivaWare™ for C Series software
Security	
Communication Interfaces	
Universal Asynchronous Receivers/Transmitter (UART)	Eight UARTs
Synchronous Serial Interface (SSI)	Four SSI modules
Inter-Integrated Circuit (I^2C)	Six I^2C modules with four transmission speeds including high-speed mode
Controller Area Network (CAN)	CAN 2.0 A/B controllers
Universal Serial Bus (USB)	USB 2.0 Device
System Integration	
Micro Direct Memory Access (µDMA)	ARM® PrimeCell® 32-channel configurable µDMA controller
General-Purpose Timer (GPTM)	Six 16/32-bit GPTM blocks and six 32/64-bit Wide GPTM blocks
Watchdog Timer (WDT)	Two watchdog timers
General-Purpose Input/Output (GPIO)	Seven physical GPIO blocks
Analog Support	
Analog-to-Digital Converter (ADC)	Two 12-bit ADC modules, each with a maximum sample rate of one million samples/second
Analog Comparator Controller	Two independent integrated analog comparators
Digital Comparator	16 digital comparators
JTAG and Serial Wire Debug (SWD)	One JTAG module with integrated ARM SWD
Package Information	
Package	64-pin LQFP
Operating Range (Ambient)	Industrial (-40°C to 85°C) temperature range

Figure 1-1 on page 40 shows the features on the TM4C1232C3PM microcontroller. Note that there are two on-chip buses that connect the core to the peripherals. The Advanced Peripheral Bus (APB) bus is the legacy bus. The Advanced High-Performance Bus (AHB) bus provides better back-to-back access performance than the APB bus.

Figure 1-1. Tiva™ TM4C1232C3PM Microcontroller High-Level Block Diagram



1.3 TM4C1232C3PM Microcontroller Features

The TM4C1232C3PM microcontroller component features and general function are discussed in more detail in the following section.

1.3.1 ARM Cortex-M4F Processor Core

All members of the Tiva™ C Series, including the TM4C1232C3PM microcontroller, are designed around an ARM Cortex-M processor core. The ARM Cortex-M processor provides the core for a high-performance, low-cost platform that meets the needs of minimal memory implementation, reduced pin count, and low power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

1.3.1.1 Processor Core (see page 58)

- 32-bit ARM Cortex-M4F architecture optimized for small-footprint embedded applications
- 80-MHz operation; 100 DMIPS performance
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set delivers the high performance expected of a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices, typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- IEEE754-compliant single-precision Floating-Point Unit (FPU)
- 16-bit SIMD vector processing unit
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast digital-signal-processing orientated multiply accumulate
- Saturating arithmetic for signal processing
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing

- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage up to specific frequencies; see "Internal Memory" on page 457 for more information.
- Ultra-low power consumption with integrated sleep modes

1.3.1.2 System Timer (SysTick) (see page 112)

ARM Cortex-M4F includes an integrated system timer, SysTick. SysTick provides a simple, 24-bit, clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used in several different ways, for example:

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

1.3.1.3 Nested Vectored Interrupt Controller (NVIC) (see page 113)

The TM4C1232C3PM controller includes the ARM Nested Vectored Interrupt Controller (NVIC). The NVIC and Cortex-M4F prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The interrupt vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, meaning that back-to-back interrupts can be performed without the overhead of state saving and restoration. Software can set eight priority levels on 7 exceptions (system handlers) and 67 interrupts.

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

1.3.1.4 System Control Block (SCB) (see page 114)

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

1.3.1.5 Memory Protection Unit (MPU) (see page 114)

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

1.3.1.6 Floating-Point Unit (FPU) (see page 119)

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

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

1.3.2 On-Chip Memory

The TM4C1232C3PM microcontroller is integrated with the following set of on-chip memory and features:

- 12 KB single-cycle SRAM
- 32 KB Flash memory
- 2KB EEPROM
- Internal ROM loaded with TivaWare™ for C Series software:
  - TivaWare™ Peripheral Driver Library
  - TivaWare Boot Loader
  - Advanced Encryption Standard (AES) cryptography tables
  - Cyclic Redundancy Check (CRC) error detection functionality

1.3.2.1 SRAM (see page 458)

The TM4C1232C3PM microcontroller provides 12 KB of single-cycle on-chip SRAM. The internal SRAM of the device is located at offset 0x2000.0000 of the device memory map.

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

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

- µDMA
- USB

1.3.2.2 Flash Memory (see page 461)

The TM4C1232C3PM microcontroller provides 32 KB of single-cycle on-chip Flash memory. The Flash memory is organized as a set of 1-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. These blocks are paired into a set of

2-KB blocks that can be individually protected. The blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

1.3.2.3 ROM (see page 459)

The TM4C1232C3PM ROM is preprogrammed with the following software and programs:

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

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

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

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

1.3.2.4 EEPROM (see page 467)

The TM4C1232C3PM microcontroller includes an EEPROM with the following features:

- 2Kbytes of memory accessible as 512 32-bit words
- 32 blocks of 16 words (64 bytes) each
- Built-in wear leveling
- Access protection per block
- Lock protection option for the whole peripheral as well as per block using 32-bit to 96-bit unlock codes (application selectable)

- Interrupt support for write completion to avoid polling
- Endurance of 500K writes (when writing at fixed offset in every alternate page in circular fashion) to 15M operations (when cycling through two pages) per each 2-page block.

1.3.3 Serial Communications Peripherals

The TM4C1232C3PM controller supports both asynchronous and synchronous serial communications with:

- CAN 2.0 A/B controller
- USB 2.0 Device
- Eight UARTs with IrDA, 9-bit and ISO 7816 support.
- Six I^{2}C modules with four transmission speeds including high-speed mode
- Four Synchronous Serial Interface modules (SSI)

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

1.3.3.1 Controller Area Network (CAN) (see page 977)

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

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

The TM4C1232C3PM microcontroller includes one CAN unit with the following features:

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

1.3.3.2 Universal Serial Bus (USB) (see page 1027)

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 TM4C1232C3PM microcontroller supports the USB 2.0 full-speed configuration in Device mode.

The USB module has the following features:

- Complies with USB-IF (Implementer's Forum) certification standards
- USB 2.0 full-speed (12 Mbps) operation with integrated PHY
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 16 endpoints
  - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
  - 7 configurable IN endpoints and 7 configurable OUT endpoints
- 4 KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - 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

1.3.3.3 UART (see page 822)

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 TM4C1232C3PM microcontroller includes eight fully programmable 16C550-type UARTs. Although the functionality is similar to a 16C550 UART, this UART design is not register compatible. The UART can generate individually masked interrupts from the Rx, Tx, modem flow control, and error conditions. The module generates a single combined interrupt when any of the interrupts are asserted and are unmasked.

The eight UARTs have the following features:

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

- Fully programmable serial interface characteristics
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no-parity bit generation/detection
  - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder/decoder providing
  - Programmable use of IrDA Serial Infrared (SIR) or UART input/output
  - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
  - Support of normal 3/16 and low-power (1.41-2.23 μs) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- Modem flow control (on UART1)
- EIA-485 9-bit support
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (μDMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level
  - Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

1.3.3.4 I^{2}C (see page 926)

The Inter-Integrated Circuit (I^{2}C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL). The I^{2}C bus interfaces to external I^{2}C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I^{2}C bus may also be used for system testing and diagnostic purposes in product development and manufacture.

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

The TM4C1232C3PM microcontroller includes six I^{2}C modules with the following features:

- Devices on the I^{2}C bus can be designated as either a master or a slave
  - Supports both transmitting and receiving data as either a master or a slave
  - Supports simultaneous master and slave operation

- Four I^{2}C modes
  - Master transmit
  - Master receive
  - Slave transmit
  - Slave receive
- Four transmission speeds:
  - Standard (100 Kbps)
  - Fast-mode (400 Kbps)
  - Fast-mode plus (1 Mbps)
  - High-speed mode (3.33 Mbps)
- Clock low timeout interrupt
- Dual slave address capability
- Glitch suppression
- 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

1.3.3.5 SSI (see page 881)

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 TM4C1232C3PM microcontroller includes four SSI modules with the following features:

- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Master or slave operation

- Programmable clock bit rate and prescaler
- Separate transmit and receive FIFOs, each 16 bits wide and 8 locations deep
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing
- Standard FIFO-based interrupts and End-of-Transmission interrupt
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted when FIFO contains 4 entries
  - Transmit single request asserted when there is space in the FIFO; burst request asserted when four or more entries are available to be written in the FIFO

1.3.4 System Integration

The TM4C1232C3PM microcontroller provides a variety of standard system functions integrated into the device, including:

- Direct Memory Access Controller (DMA)
- System control and clocks including on-chip precision 16-MHz oscillator
- Six 32-bit timers (up to twelve 16-bit), with real-time clock capability
- Six wide 64-bit timers (up to twelve 32-bit), with real-time clock capability
- Twelve 32/64-bit Capture Compare PWM (CCP) pins
- Two Watchdog Timers
  - One timer runs off the main oscillator
  - One timer runs off the precision internal oscillator
- Up to 49 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.

1.3.4.1 Direct Memory Access (see page 517)

The TM4C1232C3PM microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (µDMA). The µDMA controller provides a way to offload data transfer tasks from the Cortex-M4F processor, allowing for more efficient use of the processor and the available bus bandwidth. The µDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The µDMA controller provides the following features:

- ARM PrimeCell® 32-channel configurable µDMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of up to 256 arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
  - Independently configured and operated channels
  - Dedicated channels for supported on-chip modules
  - Flexible channel assignments
  - One channel each for receive and transmit path for bidirectional modules
  - Dedicated channel for software-initiated transfers
  - Per-channel configurable priority scheme
  - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between µDMA controller and the processor core
  - µDMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Transfer size is programmable in binary steps from 1 to 1024
- Source and destination address increment size of byte, half-word, word, or no increment
- Maskable peripheral requests
- Interrupt on transfer completion, with a separate interrupt per channel

1.3.4.2 System Control and Clocks (see page 201)

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
  - Low-power options for microcontroller: Sleep and Deep-Sleep modes with clock gating
  - Low-power options for on-chip modules: software controls shutdown of individual peripherals and memory
  - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Multiple clock sources for microcontroller system clock. The following clock sources are provided to the TM4C1232C3PM microcontroller:
  - Precision Internal Oscillator (PIOSC) providing a 16-MHz frequency
    - 16 MHz ±3% across temperature and voltage
    - Can be recalibrated with 7-bit trim resolution to achieve better accuracy (16 MHz ±1%)
    - 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.
  - Low Frequency Internal Oscillator (LFIOSC): On-chip resource used during power-saving modes
- 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

1.3.4.3 Programmable Timers (see page 636)

Programmable timers can be used to count or time external events that drive the Timer input pins. Each 16/32-bit GPTM block provides two 16-bit timers/counters that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Each 32/64-bit Wide GPTM block provides two 32-bit timers/counters that can be configured to operate independently as timersor event counters, or configured to operate as one 64-bit timer or one 64-bit Real-Time Clock (RTC). Timers can also be used to trigger analog-to-digital (ADC) conversions and DMA transfers.

The General-Purpose Timer Module (GPTM) contains six 16/32-bit GPTM blocks and six 32/64-bit Wide GPTM blocks with the following functional options:

- 16/32-bit operating modes:
  - 16- or 32-bit programmable one-shot timer

  - 16- or 32-bit programmable periodic timer
  - 16-bit general-purpose timer with an 8-bit prescaler
  - 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
  - 16-bit input-edge count- or time-capture modes with an 8-bit prescaler
  - 16-bit PWM mode with an 8-bit prescaler and software-programmable output inversion of the PWM signal

- 32/64-bit operating modes:
  - 32- or 64-bit programmable one-shot timer
  - 32- or 64-bit programmable periodic timer
  - 32-bit general-purpose timer with a 16-bit prescaler
  - 64-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
  - 32-bit input-edge count- or time-capture modes with a16-bit prescaler
  - 32-bit PWM mode with a 16-bit prescaler and software-programmable output inversion of the PWM signal
- Count up or down
- Twelve 16/32-bit Capture Compare PWM pins (CCP)
- Twelve 32/64-bit Capture Compare PWM pins (CCP)
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- Timer synchronization allows selected timers to start counting on the same clock cycle
- ADC event trigger
- User-enabled stalling when the microcontroller asserts CPU Halt flag during debug (excluding RTC mode)
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine
- Efficient transfers using Micro Direct Memory Access Controller (μDMA)
  - Dedicated channel for each timer
  - Burst request generated on timer interrupt

1.3.4.4 CCP Pins (see page 644)

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 TM4C1232C3PM microcontroller includes twelve 16/32-bit CCP pins that can be programmed to operate in the following modes:

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

1.3.4.5 Watchdog Timers (see page 706)

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 TM4C1232C3PM Watchdog Timer can generate an interrupt, a non-maskable interrupt, or a reset when a time-out value is reached. In addition, the Watchdog Timer is ARM FiRM-compliant and can be configured to generate an interrupt to the microcontroller on its first time-out, and to generate a reset signal on its second timeout. Once the Watchdog Timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

The TM4C1232C3PM microcontroller has two Watchdog Timer modules: Watchdog Timer 0 uses the system clock for its timer clock; Watchdog Timer 1 uses the PIOSC as its timer clock. The Watchdog Timer module has the following features:

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

1.3.4.6 Programmable GPIOs (see page 581)

General-purpose input/output (GPIO) pins offer flexibility for a variety of connections. The TM4C1232C3PM GPIO module is comprised of seven physical GPIO blocks, each corresponding to an individual GPIO port. The GPIO module is FiRM-compliant (compliant to the ARM Foundation IP for Real-Time Microcontrollers specification) and supports 0-49 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 1099 for the signals available to each GPIO pin).

- Up to 49 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant in input configuration
- Ports A-G accessed through the Advanced Peripheral Bus (APB)

- Fast toggle capable of a change every clock cycle for ports on AHB, every two clock cycles for ports on APB
- 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 or a µDMA transfer
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can sink 18-mA for high-current applications
  - Slew rate control for 8-mA pad drive
  - Open drain enables
  - Digital input enables

1.3.5 Analog

The TM4C1232C3PM microcontroller provides analog functions integrated into the device, including:

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

The following provides more detail on these analog functions.

1.3.5.1 ADC (see page 731)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. The TM4C1232C3PM ADC module features 12-bit conversion resolution and supports 12 input channels plus an internal temperature sensor. Four buffered sample sequencers allow rapid sampling of up to 12 analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. Each ADC module has a digital comparator function that allows the conversion value to be diverted to a comparison unit that provides eight digital comparators.

The TM4C1232C3PM microcontroller provides two ADC modules, each with the following features:

- 12 shared analog input channels

- 12-bit precision ADC
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of one million samples/second
- Optional phase shift in sample time programmable from 22.5º to 337.5º
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - GPIO
- Hardware averaging of up to 64 samples
- Eight digital comparators
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Dedicated channel for each sample sequencer
  - ADC module uses burst requests for DMA

1.3.5.2 Analog Comparators (see page 1083)

An analog comparator is a peripheral that compares two analog voltages and provides a logical output that signals the comparison result. The TM4C1232C3PM microcontroller provides two 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 TM4C1232C3PM microcontroller provides two independent integrated analog comparators with the following functions:

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

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

1.3.6 JTAG and ARM Serial Wire Debug (see page 189)

The Joint Test Action Group (JTAG) port is an IEEE standard that defines a Test Access Port and Boundary Scan Architecture for digital integrated circuits and provides a standardized serial interface for controlling the associated test logic. The TAP, Instruction Register (IR), and Data Registers (DR) can be used to test the interconnections of assembled printed circuit boards and obtain manufacturing information on the components. The JTAG Port also provides a means of accessing and controlling design-for-test features such as I/O pin observation and control, scan testing, and debugging. Texas Instruments replaces the ARM SW-DP and JTAG-DP with the ARM Serial Wire JTAG Debug Port (SWJ-DP) interface. The SWJ-DP interface combines the SWD and JTAG debug ports into one module providing all the normal JTAG debug and test functionality plus real-time access to system memory without halting the core or requiring any target resident code. The SWJ-DP interface has the following features:

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

1.3.7 Packaging and Temperature

- 64-pin RoHS-compliant LQFP package
- Industrial (-40°C to 85°C) ambient temperature range

1.4 TM4C1232C3PM Microcontroller Hardware Details

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

- “Pin Diagram” on page 1098
- “Signal Tables” on page 1099
- “Electrical Characteristics” on page 1122

- "Package Information" on page 1164

1.5 Kits

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

- Reference Design Kits accelerate product development by providing ready-to-run hardware and comprehensive documentation including hardware design files
- Evaluation Kits provide a low-cost and effective means of evaluating TM4C1232C3PM microcontrollers before purchase
- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box

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

1.6 Support Information

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

2 The Cortex-M4F Processor

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

- 32-bit ARM® Cortex™-M4F architecture optimized for small-footprint embedded applications
- 80-MHz operation; 100 DMIPS performance
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set delivers the high performance expected of a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices, typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- IEEE754-compliant single-precision Floating-Point Unit (FPU)
- 16-bit SIMD vector processing unit
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast digital-signal-processing orientated multiply accumulate
- Saturating arithmetic for signal processing
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing
- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage up to specific frequencies; see “Internal Memory” on page 457 for more information.
- Ultra-low power consumption with integrated sleep modes

The Tiva™ C Series microcontrollers builds on this core to bring high-performance 32-bit computing to

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

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

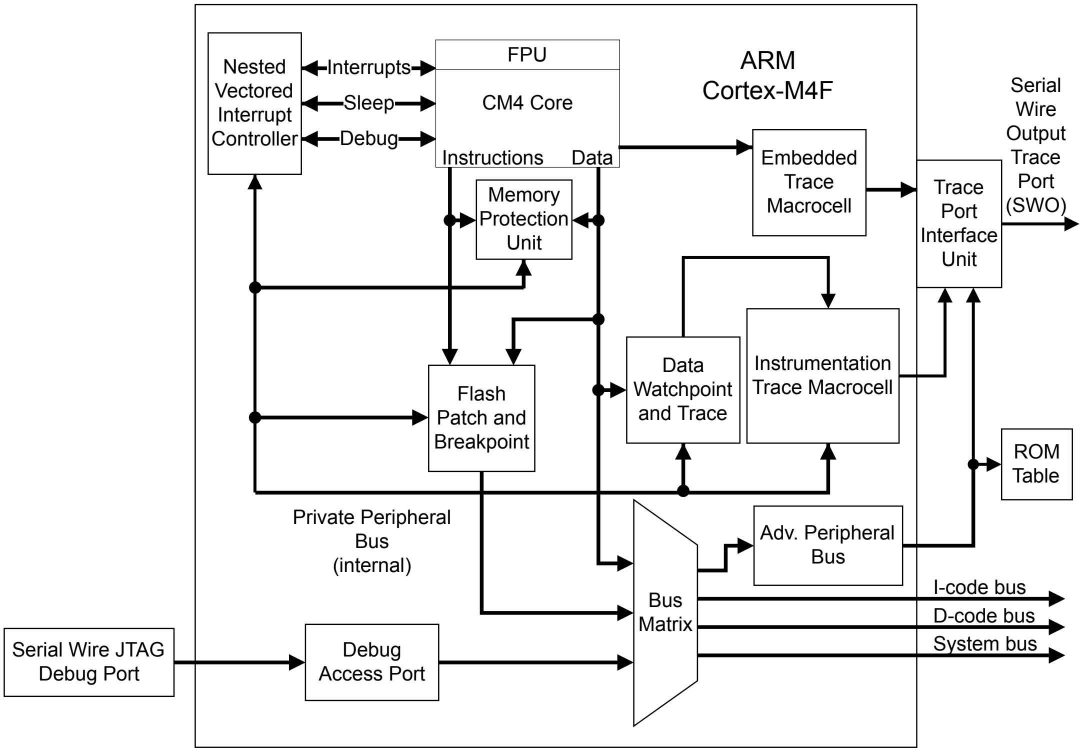
2.1 Block Diagram

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

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

The Cortex-M4F processor closely integrates a nested interrupt controller (NVIC), to deliver industry-leading interrupt performance. The TM4C1232C3PM NVIC includes a non-maskable interrupt (NMI) and provides eight interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of interrupt service routines (ISRs), dramatically reducing interrupt latency. The hardware stacking of registers and the ability to suspend load-multiple and store-multiple operations further reduce interrupt latency. Interrupt handlers do not require any assembler stubs which removes code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another. To optimize low-power designs, the NVIC integrates with the sleep modes, including Deep-sleep mode, which enables the entire device to be rapidly powered down.

Figure 2-1. CPU Block Diagram


ARM
Cortex-M4F
Nested Vectored Interrupt Controller
Interrupts
Sleep
Debug
FPU
CM4 Core
Instructions
Data
Memory Protection Unit
Embedded Trace Macrocell
Trace Port Interface Unit
Serial Wire Output Trace Port (SWO)
Flash Patch and Breakpoint
Data Watchpoint and Trace
Instrumentation Trace Macrocell
ROM Table
Private Peripheral Bus (internal)
Adv. Peripheral Bus
Bus Matrix
I-code bus
D-code bus
System bus
Serial Wire JTAG Debug Port
Debug Access Port

2.2 Overview

2.2.1 System-Level Interface

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

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

2.2.2 Integrated Configurable Debug

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

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

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

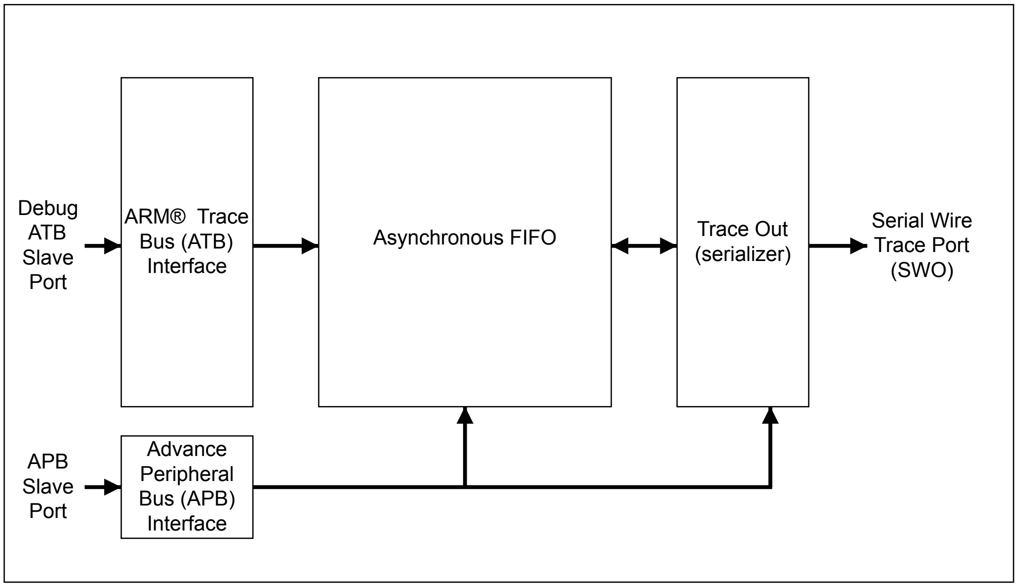
The Flash Patch and Breakpoint Unit (FPB) provides up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions for up to eight words of program code in the code memory region. This FPB enables applications stored in a read-only area of Flash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

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

2.2.3 Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the Cortex-M4F trace data from the ITM, and an off-chip Trace Port Analyzer, as shown in Figure 2-2 on page 61.

Figure 2-2. TPIU Block Diagram


Debug ATB Slave Port
ARM® Trace Bus (ATB) Interface
Asynchronous FIFO
Trace Out (serializer)
Serial Wire Trace Port (SWO)
APB Slave Port
Advance Peripheral Bus (APB) Interface

2.2.4 Cortex-M4F System Component Details

The Cortex-M4F includes the following system components:

- SysTick

  A 24-bit count-down timer that can be used as a Real-Time Operating System (RTOS) tick timer or as a simple counter (see “System Timer (SysTick)” on page 112).

- Nested Vectored Interrupt Controller (NVIC)

An embedded interrupt controller that supports low latency interrupt processing (see “Nested Vectored Interrupt Controller (NVIC)” on page 113).

- System Control Block (SCB)

  The programming model interface to the processor. The SCB provides system implementation information and system control, including configuration, control, and reporting of system exceptions (see “System Control Block (SCB)” on page 114).

- Memory Protection Unit (MPU)

  Improves system reliability by defining the memory attributes for different memory regions. The MPU provides up to eight different regions and an optional predefined background region (see “Memory Protection Unit (MPU)” on page 114).

- Floating-Point Unit (FPU)

  Fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square-root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions (see “Floating-Point Unit (FPU)” on page 119).

2.3 Programming Model

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

2.3.1 Processor Mode and Privilege Levels for Software Execution

The Cortex-M4F has two modes of operation:

- Thread mode

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

- Handler mode

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

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

- Unprivileged

  In this mode, software has the following restrictions:

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

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

  – Possibly restricted access to memory or peripherals

- Privileged

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

In Thread mode, the **CONTROL** register (see page 77) controls whether software execution is privileged or unprivileged. In Handler mode, software execution is always privileged.

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

2.3.2 Stacks

The processor uses a full descending stack, meaning that the stack pointer indicates the last stacked item on the memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks: the main stack and the process stack, with a pointer for each held in independent registers (see the **SP** register on page 67).

In Thread mode, the **CONTROL** register (see page 77) controls whether the processor uses the main stack or the process stack. In Handler mode, the processor always uses the main stack. The options for processor operations are shown in Table 2-1 on page 63.

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

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

a. See **CONTROL** (page 77).

2.3.3 Register Map

Figure 2-3 on page 64 shows the Cortex-M4F register set. Table 2-2 on page 64 lists the Core registers. The core registers are not memory mapped and are accessed by register name, so the base address is n/a (not applicable) and there is no offset.

Figure 2-3. Cortex-M4F Register Set

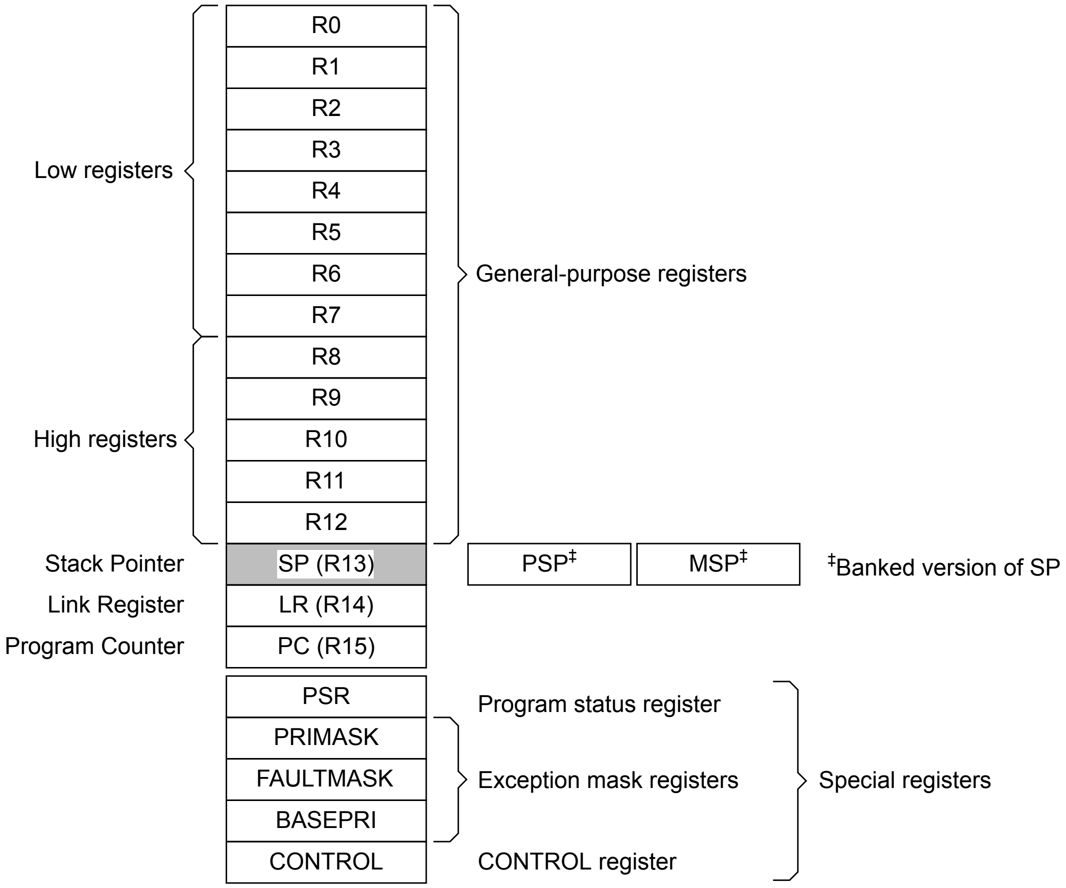


Table 2-2. Processor Register Map

Offset	Name	Type	Reset	Description	See page
-	R0	RW	-	Cortex General-Purpose Register 0	66
-	R1	RW	-	Cortex General-Purpose Register 1	66
-	R2	RW	-	Cortex General-Purpose Register 2	66
-	R3	RW	-	Cortex General-Purpose Register 3	66
-	R4	RW	-	Cortex General-Purpose Register 4	66
-	R5	RW	-	Cortex General-Purpose Register 5	66
-	R6	RW	-	Cortex General-Purpose Register 6	66
-	R7	RW	-	Cortex General-Purpose Register 7	66
-	R8	RW	-	Cortex General-Purpose Register 8	66
-	R9	RW	-	Cortex General-Purpose Register 9	66
-	R10	RW	-	Cortex General-Purpose Register 10	66
-	R11	RW	-	Cortex General-Purpose Register 11	66

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

Offset	Name	Type	Reset	Description	See page
-	R12	RW	-	Cortex General-Purpose Register 12	66
-	SP	RW	-	Stack Pointer	67
-	LR	RW	0xFFFF.FFFF	Link Register	68
-	PC	RW	-	Program Counter	69
-	PSR	RW	0x0100.0000	Program Status Register	70
-	PRIMASK	RW	0x0000.0000	Priority Mask Register	74
-	FAULTMASK	RW	0x0000.0000	Fault Mask Register	75
-	BASEPRI	RW	0x0000.0000	Base Priority Mask Register	76
-	CONTROL	RW	0x0000.0000	Control Register	77
-	FPSC	RW	-	Floating-Point Status Control	79

2.3.4 Register Descriptions

This section lists and describes the Cortex-M4F registers, in the order shown in Figure 2-3 on page 64. The core registers are not memory mapped and are accessed by register name rather than offset.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Cortex General-Purpose Register 0 (R0)

Type RW, reset -

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

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

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

Register 14: Stack Pointer (SP)

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

Stack Pointer (SP)

Type RW, reset -

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

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

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

Register 15: Link Register (LR)

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

`EXC_RETURN` is loaded into the **LR** on exception entry. See Table 2-10 on page 100 for the values and description.

Link Register (LR)

Type RW, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	LINK															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

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

Register 16: Program Counter (PC)

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

Program Counter (PC)

Type RW, reset -

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

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

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

Register 17: Program Status Register (PSR)

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

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

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

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

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

EPSR contains the Thumb state bit and the execution state bits for the If-Then (`IT`) instruction or the Interruptible-Continuable Instruction (`ICI`) field for an interrupted load multiple or store multiple instruction. Attempts to read the **EPSR** directly through application software using the `MSR` instruction always return zero. Attempts to write the **EPSR** using the `MSR` instruction in application software are always ignored. Fault handlers can examine the **EPSR** value in the stacked **PSR** to determine the operation that faulted (see “Exception Entry and Return” on page 97).

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

These registers can be accessed individually or as a combination of any two or all three registers, using the register name as an argument to the `MSR` or `MRS` instructions. For example, all of the registers can be read using **PSR** with the `MRS` instruction, or **APSR** only can be written to using **APSR** with the `MSR` instruction. page 70 shows the possible register combinations for the **PSR**. See the `MRS` and `MSR` instruction descriptions in the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information about how to access the program status registers.

Table 2-3. PSR Register Combinations

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

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

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

Program Status Register (PSR)

Type RW, reset 0x0100.0000

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

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

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

Bit/Field	Name	Type	Reset	Description
26:25	ICI / IT	RO	0x0	**EPSR** ICI / IT status These bits, along with bits 15:10, contain the Interruptible-Continuable Instruction (`ICI`) field for an interrupted load multiple or store multiple instruction or the execution state bits of the `IT` instruction. When **EPSR** holds the `ICI` execution state, bits 26:25 are zero. The If-Then block contains up to four instructions following an `IT` instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information. The value of this field is only meaningful when accessing **PSR** or **EPSR**. Note that these **EPSR** bits cannot be accessed using `MRS` and `MSR` instructions but the definitions are provided to allow the stacked (E)PSR value to be decoded within an exception handler.
24	THUMB	RO	1	**EPSR** Thumb State This bit indicates the Thumb state and should always be set. The following can clear the `THUMB` bit: ■ The `BLX`, `BX` and `POP{PC}` instructions ■ Restoration from the stacked **xPSR** value on an exception return ■ Bit 0 of the vector value on an exception entry or reset Attempting to execute instructions when this bit is clear results in a fault or lockup. See "Lockup" on page 102 for more information. The value of this bit is only meaningful when accessing **PSR** or **EPSR**.
23:20	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19:16	GE	RW	0x0	Greater Than or Equal Flags See the description of the `SEL` instruction in the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information. The value of this field is only meaningful when accessing **PSR** or **APSR**.

Bit/Field	Name	Type	Reset	Description
15:10	ICI / IT	RO	0x0	**EPSR** ICI / IT status These bits, along with bits 26:25, contain the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction or the execution state bits of the IT instruction. When an interrupt occurs during the execution of an LDM, STM, PUSH POP, VLDM, VSTM, VPUSH, or VPOP instruction, the processor stops the load multiple or store multiple instruction operation temporarily and stores the next register operand in the multiple operation to bits 15:12. After servicing the interrupt, the processor returns to the register pointed to by bits 15:12 and resumes execution of the multiple load or store instruction. When **EPSR** holds the ICI execution state, bits 11:10 are zero. The If-Then block contains up to four instructions following a 16-bit IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information. The value of this field is only meaningful when accessing **PSR** or **EPSR**.
9:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ISRNUM	RO	0x00	**IPSR** ISR Number This field contains the exception type number of the current Interrupt Service Routine (ISR).

Value	Description
0x00	Thread mode
0x01	Reserved
0x02	NMI
0x03	Hard fault
0x04	Memory management fault
0x05	Bus fault
0x06	Usage fault
0x07-0x0A	Reserved
0x0B	SVCall
0x0C	Reserved for Debug
0x0D	Reserved
0x0E	PendSV
0x0F	SysTick
0x10	Interrupt Vector 0
0x11	Interrupt Vector 1
...	...
0x9A	Interrupt Vector 138

See “Exception Types” on page 91 for more information.

The value of this field is only meaningful when accessing **PSR** or **IPSR**.

Register 18: Priority Mask Register (PRIMASK)

The **PRIMASK** register prevents activation of all exceptions with programmable priority. Reset, non-maskable interrupt (NMI), and hard fault are the only exceptions with fixed priority. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the **PRIMASK** register, and the CPS instruction may be used to change the value of the **PRIMASK** register. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 91.

Priority Mask Register (PRIMASK)

Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															PRIMASK
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

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

Register 19: Fault Mask Register (FAULTMASK)

The **FAULTMASK** register prevents activation of all exceptions except for the Non-Maskable Interrupt (NMI). Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The `MSR` and `MRS` instructions are used to access the **FAULTMASK** register, and the `CPS` instruction may be used to change the value of the **FAULTMASK** register. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information on these instructions. For more information on exception priority levels, see "Exception Types" on page 91.

Fault Mask Register (FAULTMASK)

Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															FAULTMASK
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Value	Description
1	Prevents the activation of all exceptions except for NMI.
0	No effect.

The processor clears the `FAULTMASK` bit on exit from any exception handler except the NMI handler.

Register 20: Base Priority Mask Register (BASEPRI)

The **BASEPRI** register defines the minimum priority for exception processing. When **BASEPRI** is set to a nonzero value, it prevents the activation of all exceptions with the same or lower priority level as the **BASEPRI** value. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. For more information on exception priority levels, see “Exception Types” on page 91.

Base Priority Mask Register (BASEPRI)

Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								BASEPRI			reserved				
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	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.
7:5	BASEPRI	RW	0x0	Base Priority Any exception that has a programmable priority level with the same or lower priority as the value of this field is masked. The **PRIMASK** register can be used to mask all exceptions with programmable priority levels. Higher priority exceptions have lower priority levels. Value Description 0x0 All exceptions are unmasked. 0x1 All exceptions with priority level 1-7 are masked. 0x2 All exceptions with priority level 2-7 are masked. 0x3 All exceptions with priority level 3-7 are masked. 0x4 All exceptions with priority level 4-7 are masked. 0x5 All exceptions with priority level 5-7 are masked. 0x6 All exceptions with priority level 6-7 are masked. 0x7 All exceptions with priority level 7 are masked.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 21: Control Register (CONTROL)

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

Handler mode always uses the **MSP**, so the processor ignores explicit writes to the `ASP` bit of the **CONTROL** register when in Handler mode. The exception entry and return mechanisms automatically update the **CONTROL** register based on the EXC_RETURN value (see Table 2-10 on page 100). In an OS environment, threads running in Thread mode should use the process stack and the kernel and exception handlers should use the main stack. By default, Thread mode uses the **MSP**. To switch the stack pointer used in Thread mode to the **PSP**, either use the `MSR` instruction to set the `ASP` bit, as detailed in the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)*, or perform an exception return to Thread mode with the appropriate EXC_RETURN value, as shown in Table 2-10 on page 100.

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

Control Register (CONTROL)

Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													FPCA	ASP	TMPL
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FPCA	RW	0	Floating-Point Context Active

Value	Description
1	Floating-point context active
0	No floating-point context active

The Cortex-M4F uses this bit to determine whether to preserve floating-point state when processing an exception.

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

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

Register 22: Floating-Point Status Control (FPSC)

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

Floating-Point Status Control (FPSC)

Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	N	Z	C	V	reserved	AHP	DN	FZ	RMODE		reserved					
Type	RW	RW	RW	RW	RO	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO
Reset	-	-	-	-	0	-	-	-	-	-	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								IDC	reserved		IXC	UFC	OFC	DZC	IOC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RO	RO	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	0	0	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31	N	RW	-	Negative Condition Code Flag Floating-point comparison operations update this condition code flag.
30	Z	RW	-	Zero Condition Code Flag Floating-point comparison operations update this condition code flag.
29	C	RW	-	Carry Condition Code Flag Floating-point comparison operations update this condition code flag.
28	V	RW	-	Overflow Condition Code Flag Floating-point comparison operations update this condition code flag.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	AHP	RW	-	Alternative Half-Precision When set, alternative half-precision format is selected. When clear, IEEE half-precision format is selected. The AHP bit in the **FPDSC** register holds the default value for this bit.
25	DN	RW	-	Default NaN Mode When set, any operation involving one or more NaNs returns the Default NaN. When clear, NaN operands propagate through to the output of a floating-point operation. The DN bit in the **FPDSC** register holds the default value for this bit.
24	FZ	RW	-	Flush-to-Zero Mode When set, Flush-to-Zero mode is enabled. When clear, Flush-to-Zero mode is disabled and the behavior of the floating-point system is fully compliant with the IEEE 754 standard. The FZ bit in the **FPDSC** register holds the default value for this bit.

Bit/Field	Name	Type	Reset	Description
23:22	RMODE	RW	-	Rounding Mode The specified rounding mode is used by almost all floating-point instructions. The `RMODE` bit in the **FPDSC** register holds the default value for this bit. Value Description 0x0 Round to Nearest (RN) mode 0x1 Round towards Plus Infinity (RP) mode 0x2 Round towards Minus Infinity (RM) mode 0x3 Round towards Zero (RZ) mode
21:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	IDC	RW	-	Input Denormal Cumulative Exception When set, indicates this exception has occurred since 0 was last written to this bit.
6:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	IXC	RW	-	Inexact Cumulative Exception When set, indicates this exception has occurred since 0 was last written to this bit.
3	UFC	RW	-	Underflow Cumulative Exception When set, indicates this exception has occurred since 0 was last written to this bit.
2	OFC	RW	-	Overflow Cumulative Exception When set, indicates this exception has occurred since 0 was last written to this bit.
1	DZC	RW	-	Division by Zero Cumulative Exception When set, indicates this exception has occurred since 0 was last written to this bit.
0	IOC	RW	-	Invalid Operation Cumulative Exception When set, indicates this exception has occurred since 0 was last written to this bit.

2.3.5 Exceptions and Interrupts

The Cortex-M4F processor supports interrupts and system exceptions. The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses Handler mode to handle all exceptions except for reset. See “Exception Entry and Return” on page 97 for more information.

The NVIC registers control interrupt handling. See “Nested Vectored Interrupt Controller (NVIC)” on page 113 for more information.

2.3.6 Data Types

The Cortex-M4F supports 32-bit words, 16-bit halfwords, and 8-bit bytes. The processor also supports 64-bit data transfer instructions. All instruction and data memory accesses are little endian. See “Memory Regions, Types and Attributes” on page 83 for more information.

2.4 Memory Model

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

The memory map for the TM4C1232C3PM controller is provided in Table 2-4 on page 81. In this manual, register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map.

The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data (see “Bit-Banding” on page 86).

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers (see “Cortex-M4 Peripherals” on page 111).

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

Table 2-4. Memory Map

Start	End	Description	For details, see page ...
Memory			
0x0000.0000	0x0000.7FFF	On-chip Flash	472
0x0000.8000	0x1FFF.FFFF	Reserved	-
0x2000.0000	0x2000.2FFF	Bit-banded on-chip SRAM	458
0x2000.3000	0x21FF.FFFF	Reserved	-
0x2200.0000	0x2205.FFFF	Bit-band alias of bit-banded on-chip SRAM starting at 0x2000.0000	458
0x2206.0000	0x3FFF.FFFF	Reserved	-
Peripherals			
0x4000.0000	0x4000.0FFF	Watchdog timer 0	708
0x4000.1000	0x4000.1FFF	Watchdog timer 1	708
0x4000.2000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	590
0x4000.5000	0x4000.5FFF	GPIO Port B	590

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

Start	End	Description	For details, see page ...
0x4000.6000	0x4000.6FFF	GPIO Port C	590
0x4000.7000	0x4000.7FFF	GPIO Port D	590
0x4000.8000	0x4000.8FFF	SSI0	896
0x4000.9000	0x4000.9FFF	SSI1	896
0x4000.A000	0x4000.AFFF	SSI2	896
0x4000.B000	0x4000.BFFF	SSI3	896
0x4000.C000	0x4000.CFFF	UART0	832
0x4000.D000	0x4000.DFFF	UART1	832
0x4000.E000	0x4000.EFFF	UART2	832
0x4000.F000	0x4000.FFFF	UART3	832
0x4001.0000	0x4001.0FFF	UART4	832
0x4001.1000	0x4001.1FFF	UART5	832
0x4001.2000	0x4001.2FFF	UART6	832
0x4001.3000	0x4001.3FFF	UART7	832
0x4001.4000	0x4001.FFFF	Reserved	-
Peripherals			
0x4002.0000	0x4002.0FFF	I^2C 0	946
0x4002.1000	0x4002.1FFF	I^2C 1	946
0x4002.2000	0x4002.2FFF	I^2C 2	946
0x4002.3000	0x4002.3FFF	I^2C 3	946
0x4002.4000	0x4002.4FFF	GPIO Port E	590
0x4002.5000	0x4002.5FFF	GPIO Port F	590
0x4002.6000	0x4002.6FFF	GPIO Port G	590
0x4002.7000	0x4002.FFFF	Reserved	-
0x4003.0000	0x4003.0FFF	16/32-bit Timer 0	657
0x4003.1000	0x4003.1FFF	16/32-bit Timer 1	657
0x4003.2000	0x4003.2FFF	16/32-bit Timer 2	657
0x4003.3000	0x4003.3FFF	16/32-bit Timer 3	657
0x4003.4000	0x4003.4FFF	16/32-bit Timer 4	657
0x4003.5000	0x4003.5FFF	16/32-bit Timer 5	657
0x4003.6000	0x4003.6FFF	32/64-bit Timer 0	657
0x4003.7000	0x4003.7FFF	32/64-bit Timer 1	657
0x4003.8000	0x4003.8FFF	ADC0	750
0x4003.9000	0x4003.9FFF	ADC1	750
0x4003.A000	0x4003.BFFF	Reserved	-
0x4003.C000	0x4003.CFFF	Analog Comparators	1088
0x4003.D000	0x4003.FFFF	Reserved	-
0x4004.0000	0x4004.0FFF	CAN0 Controller	996
0x4004.1000	0x4004.BFFF	Reserved	-
0x4004.C000	0x4004.CFFF	32/64-bit Timer 2	657
0x4004.D000	0x4004.DFFF	32/64-bit Timer 3	657

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

Start	End	Description	For details, see page ...
0x4004.E000	0x4004.EFFF	32/64-bit Timer 4	657
0x4004.F000	0x4004.FFFF	32/64-bit Timer 5	657
0x4005.0000	0x4005.0FFF	USB	1035
0x4005.1000	0x4005.7FFF	Reserved	-
0x4005.8000	0x4005.8FFF	GPIO Port A (AHB aperture)	590
0x4005.9000	0x4005.9FFF	GPIO Port B (AHB aperture)	590
0x4005.A000	0x4005.AFFF	GPIO Port C (AHB aperture)	590
0x4005.B000	0x4005.BFFF	GPIO Port D (AHB aperture)	590
0x4005.C000	0x4005.CFFF	GPIO Port E (AHB aperture)	590
0x4005.D000	0x4005.DFFF	GPIO Port F (AHB aperture)	590
0x4005.E000	0x4005.EFFF	GPIO Port G (AHB aperture)	590
0x4005.F000	0x400A.EFFF	Reserved	-
0x400A.F000	0x400A.FFFF	EEPROM and Key Locker	472
0x400B.0000	0x400B.FFFF	Reserved	-
0x400C.0000	0x400C.0FFF	I^{2}C 4	946
0x400C.1000	0x400C.1FFF	I^{2}C 5	946
0x400C.2000	0x400F.8FFF	Reserved	-
0x400F.9000	0x400F.9FFF	System Exception Module	449
0x400F.A000	0x400F.CFFF	Reserved	-
0x400F.D000	0x400F.DFFF	Flash memory control	472
0x400F.E000	0x400F.EFFF	System control	219
0x400F.F000	0x400F.FFFF	μDMA	538
0x4010.0000	0x41FF.FFFF	Reserved	-
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-
0x4400.0000	0xDFFF.FFFF	Reserved	-
Private Peripheral Bus			
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	60
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	60
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	60
0xE000.3000	0xE000.DFFF	Reserved	-
0xE000.E000	0xE000.EFFF	Cortex-M4F Peripherals (SysTick, NVIC, MPU, FPU and SCB)	123
0xE000.F000	0xE003.FFFF	Reserved	-
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	61
0xE004.1000	0xE004.1FFF	Embedded Trace Macrocell (ETM)	60
0xE004.2000	0xFFFF.FFFF	Reserved	-

2.4.1 Memory Regions, Types and Attributes

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

The memory types are:

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

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

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

2.4.2 Memory System Ordering of Memory Accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing the order does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions (see “Software Ordering of Memory Accesses” on page 85).

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

2.4.3 Behavior of Memory Accesses

Table 2-5 on page 84 shows the behavior of accesses to each region in the memory map. See “Memory Regions, Types and Attributes” on page 83 for more information on memory types and the XN attribute. Tiva™ C Series devices may have reserved memory areas within the address ranges shown below (refer to Table 2-4 on page 81 for more information).

Table 2-5. Memory Access Behavior

Address Range	Memory Region	Memory Type	Execute Never (XN)	Description
0x0000.0000 - 0x1FFF.FFFF	Code	Normal	-	This executable region is for program code. Data can also be stored here.
0x2000.0000 - 0x3FFF.FFFF	SRAM	Normal	-	This executable region is for data. Code can also be stored here. This region includes bit band and bit band alias areas (see Table 2-6 on page 86).
0x4000.0000 - 0x5FFF.FFFF	Peripheral	Device	XN	This region includes bit band and bit band alias areas (see Table 2-7 on page 86).
0x6000.0000 - 0x9FFF.FFFF	External RAM	Normal	-	This executable region is for data.
0xA000.0000 - 0xDFFF.FFFF	External device	Device	XN	This region is for external device memory.
0xE000.0000- 0xE00F.FFFF	Private peripheral bus	Strongly Ordered	XN	This region includes the NVIC, system timer, and system control block.
0xE010.0000- 0xFFFF.FFFF	Reserved	-	-	-

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

The MPU can override the default memory access behavior described in this section. For more information, see "Memory Protection Unit (MPU)" on page 114.

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

2.4.4 Software Ordering of Memory Accesses

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

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

"Memory System Ordering of Memory Accesses" on page 84 describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The Cortex-M4F has the following memory barrier instructions:

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

Memory barrier instructions can be used in the following situations:

- MPU programming
  - If the MPU settings are changed and the change must be effective on the very next instruction, use a `DSB` instruction to ensure the effect of the MPU takes place immediately at the end of context switching.
  - Use an `ISB` instruction to ensure the new MPU setting takes effect immediately after programming the MPU region or regions, if the MPU configuration code was accessed using a branch or call. If the MPU configuration code is entered using exception mechanisms, then an `ISB` instruction is not required.
- Vector table

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

- Self-modifying code

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

- Memory map switching

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

- Dynamic exception priority change

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

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

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

2.4.5 Bit-Banding

A bit-band region maps each word in a bit-band alias region to a single bit in the bit-band region. The bit-band regions occupy the lowest 1 MB of the SRAM and peripheral memory regions. Accesses to the 32-MB SRAM alias region map to the 1-MB SRAM bit-band region, as shown in Table 2-6 on page 86. Accesses to the 32-MB peripheral alias region map to the 1-MB peripheral bit-band region, as shown in Table 2-7 on page 86. For the specific address range of the bit-band regions, see Table 2-4 on page 81.

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

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

Table 2-6. SRAM Memory Bit-Banding Regions

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x2000.0000	0x2000.2FFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias.
0x2200.0000	0x2205.FFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

Table 2-7. Peripheral Memory Bit-Banding Regions

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x4000.0000	0x400F.FFFF	Peripheral bit-band region	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias.

Table 2-7. Peripheral Memory Bit-Banding Regions *(continued)*

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x4200.0000	0x43FF.FFFF	Peripheral bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted.

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

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

```
bit_word_addr = bit_band_base + bit_word_offset
```

where:

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

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

bit_band_base
The starting address of the alias region.

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

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

Figure 2-4 on page 88 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

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

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

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

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

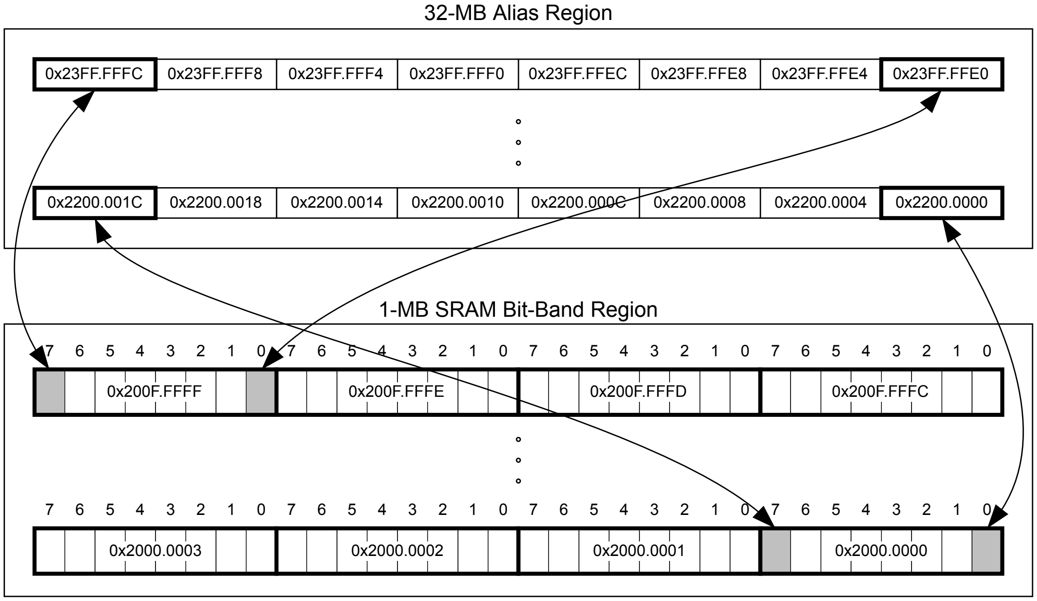
- The alias word at 0x2200.0000 maps to bit 0 of the bit-band byte at 0x2000.0000:

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

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

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

Figure 2-4. Bit-Band Mapping



2.4.5.1 Directly Accessing an Alias Region

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

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

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

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

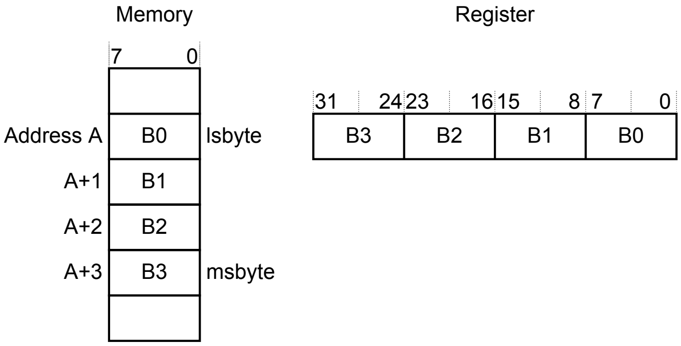
2.4.5.2 Directly Accessing a Bit-Band Region

"Behavior of Memory Accesses" on page 84 describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

2.4.6 Data Storage

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. Data is stored in little-endian format, with the least-significant byte (lsbyte) of a word stored at the lowest-numbered byte, and the most-significant byte (msbyte) stored at the highest-numbered byte. Figure 2-5 on page 89 illustrates how data is stored.

Figure 2-5. Data Storage



2.4.7 Synchronization Primitives

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

A pair of synchronization primitives consists of:

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

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

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

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

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

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

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

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

1. Use a Load-Exclusive instruction to read from the semaphore address to check whether the semaphore is free.

2. If the semaphore is free, use a Store-Exclusive to write the claim value to the semaphore address.

3. If the returned status bit from step 2 indicates that the Store-Exclusive succeeded, then the software has claimed the semaphore. However, if the Store-Exclusive failed, another process might have claimed the semaphore after the software performed step 1.

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

- It executes a `CLREX` instruction.

- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.

- An exception occurs, which means the processor can resolve semaphore conflicts between different threads.

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

2.5 Exception Model

The ARM Cortex-M4F processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration.

Table 2-8 on page 92 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 67 interrupts (listed in Table 2-9 on page 93).

Priorities on the system handlers are set with the NVIC **System Handler Priority n (SYSPRIn)** registers. Interrupts are enabled through the NVIC **Interrupt Set Enable n (ENn)** register and prioritized with the NVIC **Interrupt Priority n (PRIn)** registers. Priorities can be grouped by splitting priority levels into preemption priorities and subpriorities. All the interrupt registers are described in "Nested Vectored Interrupt Controller (NVIC)" on page 113.

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

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

See "Nested Vectored Interrupt Controller (NVIC)" on page 113 for more information on exceptions and interrupts.

2.5.1 Exception States

Each exception is in one of the following states:

- **Inactive.** The exception is not active and not pending.
- **Pending.** The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.
- **Active.** An exception that is being serviced by the processor but has not completed.

  Note: An exception handler can interrupt the execution of another exception handler. In this case, both exceptions are in the active state.
- **Active and Pending.** The exception is being serviced by the processor, and there is a pending exception from the same source.

2.5.2 Exception Types

The exception types are:

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

– An error on exception return

An unaligned address on a word or halfword memory access or division by zero can cause a usage fault when the core is properly configured.

- **SVCall.** A supervisor call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.
- **Debug Monitor.** This exception is caused by the debug monitor (when not halting). This exception is only active when enabled. This exception does not activate if it is a lower priority than the current activation.
- **PendSV.** PendSV is a pendable, interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active. PendSV is triggered using the **Interrupt Control and State (INTCTRL)** register.
- **SysTick.** A SysTick exception is an exception that the system timer generates when it reaches zero when it is enabled to generate an interrupt. Software can also generate a SysTick exception using the **Interrupt Control and State (INTCTRL)** register. In an OS environment, the processor can use this exception as system tick.
- **Interrupt (IRQ).** An interrupt, or IRQ, is an exception signaled by a peripheral or generated by a software request and fed through the NVIC (prioritized). All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor. Table 2-9 on page 93 lists the interrupts on the TM4C1232C3PM controller.

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

Privileged software can disable the exceptions that Table 2-8 on page 92 shows as having configurable priority (see the **SYSHNDCTRL** register on page 162 and the **DIS0** register on page 133).

For more information about hard faults, memory management faults, bus faults, and usage faults, see “Fault Handling” on page 100.

Table 2-8. Exception Types

Exception Type	Vector Number	Priority[a]	Vector Address or Offset[b]	Activation
-	0	-	0x0000.0000	Stack top is loaded from the first entry of the vector table on reset.
Reset	1	-3 (highest)	0x0000.0004	Asynchronous
Non-Maskable Interrupt (NMI)	2	-2	0x0000.0008	Asynchronous
Hard Fault	3	-1	0x0000.000C	-
Memory Management	4	programmable[c]	0x0000.0010	Synchronous
Bus Fault	5	programmable[c]	0x0000.0014	Synchronous when precise and asynchronous when imprecise
Usage Fault	6	programmable[c]	0x0000.0018	Synchronous
-	7-10	-	-	Reserved
SVCall	11	programmable[c]	0x0000.002C	Synchronous
Debug Monitor	12	programmable[c]	0x0000.0030	Synchronous
-	13	-	-	Reserved

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

Exception Type	Vector Number	Priority[a]	Vector Address or Offset[b]	Activation
PendSV	14	programmable[c]	0x0000.0038	Asynchronous
SysTick	15	programmable[c]	0x0000.003C	Asynchronous
Interrupts	16 and above	programmable[d]	0x0000.0040 and above	Asynchronous

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

b. See “Vector Table” on page 95.

c. See **SYSPRI1** on page 159.

d. See **PRIn** registers on page 141.

Table 2-9. Interrupts

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
0-15	-	0x0000.0000 - 0x0000.003C	Processor exceptions
16	0	0x0000.0040	GPIO Port A
17	1	0x0000.0044	GPIO Port B
18	2	0x0000.0048	GPIO Port C
19	3	0x0000.004C	GPIO Port D
20	4	0x0000.0050	GPIO Port E
21	5	0x0000.0054	UART0
22	6	0x0000.0058	UART1
23	7	0x0000.005C	SSI0
24	8	0x0000.0060	I^{2}C0
25-29	9-13	-	Reserved
30	14	0x0000.0078	ADC0 Sequence 0
31	15	0x0000.007C	ADC0 Sequence 1
32	16	0x0000.0080	ADC0 Sequence 2
33	17	0x0000.0084	ADC0 Sequence 3
34	18	0x0000.0088	Watchdog Timers 0 and 1
35	19	0x0000.008C	16/32-Bit Timer 0A
36	20	0x0000.0090	16/32-Bit Timer 0B
37	21	0x0000.0094	16/32-Bit Timer 1A
38	22	0x0000.0098	16/32-Bit Timer 1B
39	23	0x0000.009C	16/32-Bit Timer 2A
40	24	0x0000.00A0	16/32-Bit Timer 2B
41	25	0x0000.00A4	Analog Comparator 0
42	26	0x0000.00A8	Analog Comparator 1
43	27	-	Reserved
44	28	0x0000.00B0	System Control
45	29	0x0000.00B4	Flash Memory Control and EEPROM Control
46	30	0x0000.00B8	GPIO Port F
47	31	0x0000.00BC	GPIO Port G
48	32	-	Reserved

Table 2-9. Interrupts *(continued)*

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
49	33	0x0000.00C4	UART2
50	34	0x0000.00C8	SSI1
51	35	0x0000.00CC	16/32-Bit Timer 3A
52	36	0x0000.00D0	16/32-Bit Timer 3B
53	37	0x0000.00D4	I^{2}C1
54	38	-	Reserved
55	39	0x0000.00DC	CAN0
56-59	40-43	-	Reserved
60	44	0x0000.00F0	USB
61	45	-	Reserved
62	46	0x0000.00F8	µDMA Software
63	47	0x0000.00FC	µDMA Error
64	48	0x0000.0100	ADC1 Sequence 0
65	49	0x0000.0104	ADC1 Sequence 1
66	50	0x0000.0108	ADC1 Sequence 2
67	51	0x0000.010C	ADC1 Sequence 3
68-72	52-56	-	Reserved
73	57	0x0000.0124	SSI2
74	58	0x0000.0128	SSI3
75	59	0x0000.012C	UART3
76	60	0x0000.0130	UART4
77	61	0x0000.0134	UART5
78	62	0x0000.0138	UART6
79	63	0x0000.013C	UART7
80-83	64-67	0x0000.0140 - 0x0000.014C	Reserved
84	68	0x0000.0150	I^{2}C2
85	69	0x0000.0154	I^{2}C3
86	70	0x0000.0158	16/32-Bit Timer 4A
87	71	0x0000.015C	16/32-Bit Timer 4B
88-107	72-91	0x0000.0160 - 0x0000.01AC	Reserved
108	92	0x0000.01B0	16/32-Bit Timer 5A
109	93	0x0000.01B4	16/32-Bit Timer 5B
110	94	0x0000.01B8	32/64-Bit Timer 0A
111	95	0x0000.01BC	32/64-Bit Timer 0B
112	96	0x0000.01C0	32/64-Bit Timer 1A
113	97	0x0000.01C4	32/64-Bit Timer 1B
114	98	0x0000.01C8	32/64-Bit Timer 2A
115	99	0x0000.01CC	32/64-Bit Timer 2B
116	100	0x0000.01D0	32/64-Bit Timer 3A
117	101	0x0000.01D4	32/64-Bit Timer 3B

Table 2-9. Interrupts *(continued)*

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
118	102	0x0000.01D8	32/64-Bit Timer 4A
119	103	0x0000.01DC	32/64-Bit Timer 4B
120	104	0x0000.01E0	32/64-Bit Timer 5A
121	105	0x0000.01E4	32/64-Bit Timer 5B
122	106	0x0000.01E8	System Exception (imprecise)
123-124	107-108	-	Reserved
125	109	0x0000.01F4	I^{2}C4
126	110	0x0000.01F8	I^{2}C5
127-154	111-138	-	Reserved

2.5.3 Exception Handlers

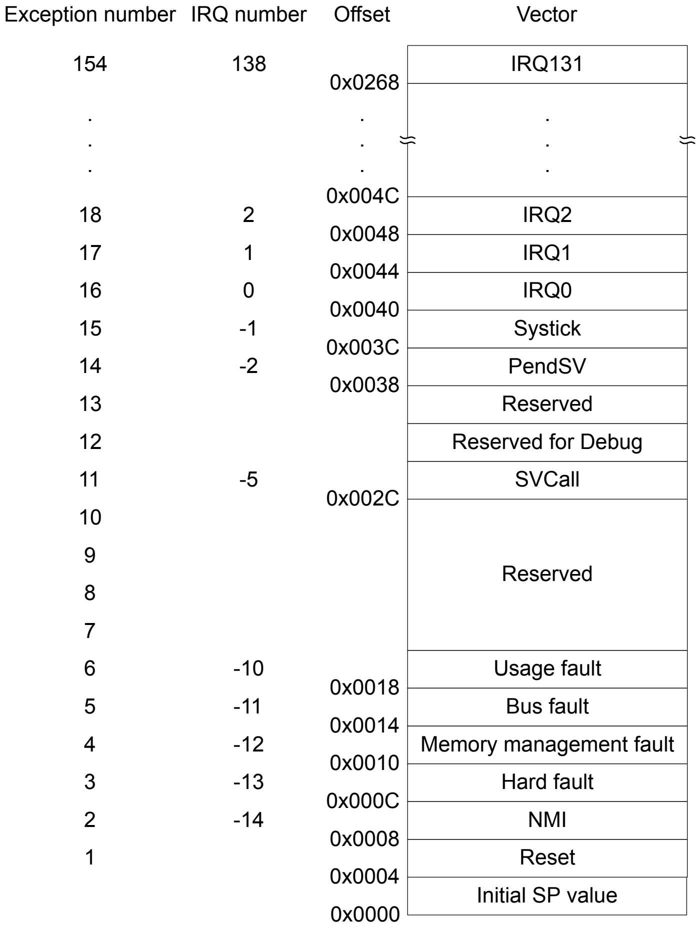
The processor handles exceptions using:

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

2.5.4 Vector Table

The vector table contains the reset value of the stack pointer and the start addresses, also called exception vectors, for all exception handlers. The vector table is constructed using the vector address or offset shown in Table 2-8 on page 92. Figure 2-6 on page 96 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code

Figure 2-6. Vector Table



Exception number	IRQ number	Offset	Vector
154	138	0x0268	IRQ131
.		.	.
.		.	.
.		.	.
		0x004C	
18	2	0x0048	IRQ2
17	1	0x0044	IRQ1
16	0	0x0040	IRQ0
15	-1	0x003C	Systick
14	-2	0x0038	PendSV
13			Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10 9 8 7			Reserved
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

On system reset, the vector table is fixed at address 0x0000.0000. Privileged software can write to the **Vector Table Offset (VTABLE)** register to relocate the vector table start address to a different memory location, in the range 0x0000.0400 to 0x3FFF.FC00 (see "Vector Table" on page 95). Note that when configuring the **VTABLE** register, the offset must be aligned on a 1024-byte boundary.

2.5.5 Exception Priorities

As Table 2-8 on page 92 shows, all exceptions have an associated priority, with a lower priority value indicating a higher priority and configurable priorities for all exceptions except Reset, Hard fault, and NMI. If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities, see page 159 and page 141.

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

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

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

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

2.5.6 Interrupt Priority Grouping

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

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

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

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

For information about splitting the interrupt priority fields into group priority and subpriority, see page 153.

2.5.7 Exception Entry and Return

Descriptions of exception handling use the following terms:

- **Preemption.** When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See “Interrupt Priority Grouping” on page 97 for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See “Exception Entry” on page 98 more information.

- **Return.** Return occurs when the exception handler is completed, and there is no pending exception with sufficient priority to be serviced and the completed exception handler was not handling a late-arriving exception. The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See “Exception Return” on page 99 for more information.

- **Tail-Chaining.** This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.

- **Late-Arriving.** This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore, the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

2.5.7.1 Exception Entry

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

When one exception preempts another, the exceptions are nested.

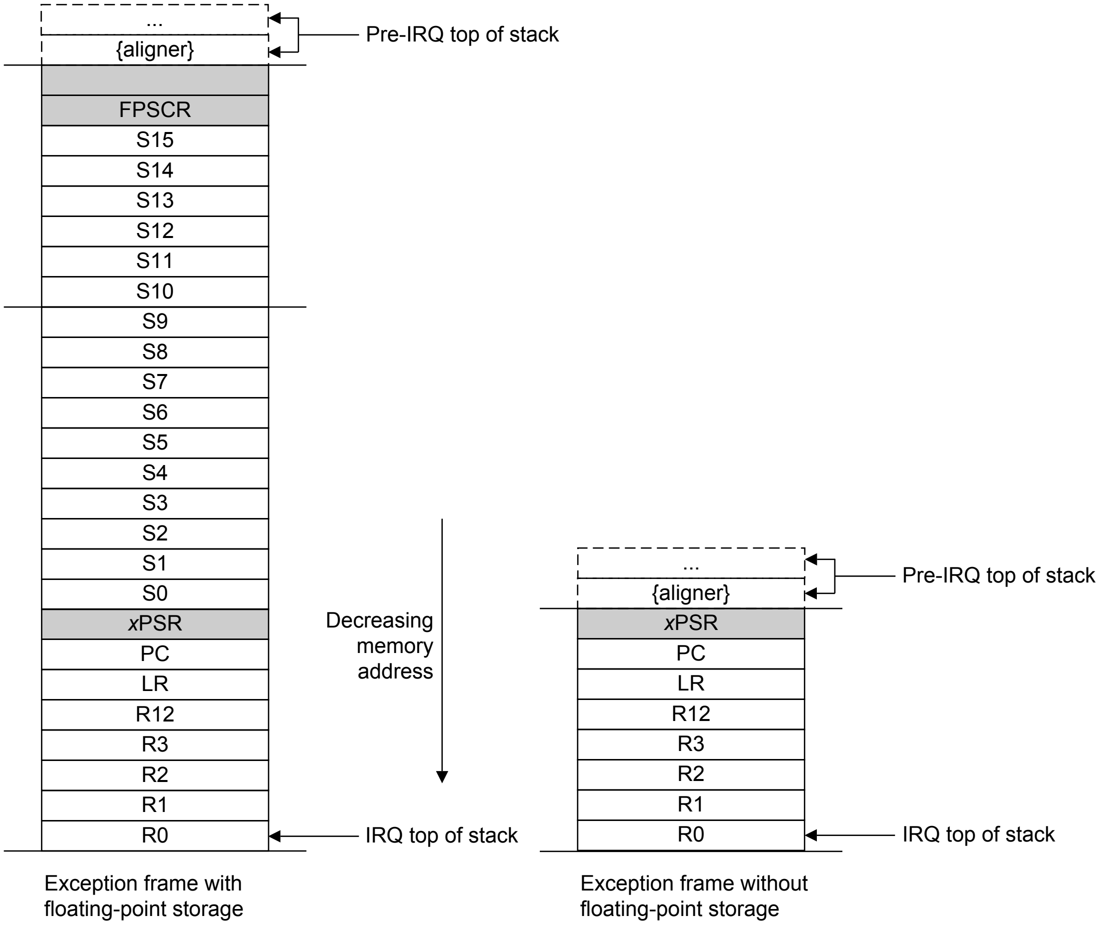
Sufficient priority means the exception has more priority than any limits set by the mask registers (see **PRIMASK** on page 74, **FAULTMASK** on page 75, and **BASEPRI** on page 76). An exception with less priority than this is pending but is not handled by the processor.

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

When using floating-point routines, the Cortex-M4F processor automatically stacks the architected floating-point state on exception entry. Figure 2-7 on page 99 shows the Cortex-M4F stack frame layout when floating-point state is preserved on the stack as the result of an interrupt or an exception.

Note: Where stack space for floating-point state is not allocated, the stack frame is the same as that of ARMv7-M implementations without an FPU. Figure 2-7 on page 99 shows this stack frame also.

Figure 2-7. Exception Stack Frame



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

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

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

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

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

2.5.7.2 Exception Return

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

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

EXC_RETURN is the value loaded into the **LR** on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest five bits of this value provide information on the return stack and processor mode. Table 2-10 on page 100 shows the EXC_RETURN values with a description of the exception return behavior.

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

Table 2-10. Exception Return Behavior

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

2.6 Fault Handling

Faults are a subset of the exceptions (see “Exception Model” on page 90). The following conditions generate a fault:

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

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

2.6.1 Fault Types

Table 2-11 on page 101 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates the fault has occurred. See page 166 for more information about the fault status registers.

Table 2-11. Faults

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

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

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

2.6.2 Fault Escalation and Hard Faults

All fault exceptions except for hard fault have configurable exception priority (see **SYSPRI1** on page 159). Software can disable execution of the handlers for these faults (see **SYSHNDCTRL** on page 162).

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler as described in “Exception Model” on page 90.

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

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

If a bus fault occurs during a stack push when entering a bus fault handler, the bus fault does not escalate to a hard fault. Thus if a corrupted stack causes a fault, the fault handler executes even though the stack push for the handler failed. The fault handler operates but the stack contents are corrupted.

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

2.6.3 Fault Status Registers and Fault Address Registers

The fault status registers indicate the cause of a fault. For bus faults and memory management faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in Table 2-12 on page 102.

Table 2-12. Fault Status and Fault Address Registers

Handler	Status Register Name	Address Register Name	Register Description
Hard fault	**Hard Fault Status (HFAULTSTAT)**	-	page 172
Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	**Memory Management Fault Address (MMADDR)**	page 166 page 173
Bus fault	**Bus Fault Status (BFAULTSTAT)**	**Bus Fault Address (FAULTADDR)**	page 166 page 174
Usage fault	**Usage Fault Status (UFAULTSTAT)**	-	page 166

2.6.4 Lockup

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

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

2.7 Power Management

The Cortex-M4F processor sleep modes reduce power consumption:

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

The SLEEPDEEP bit of the **System Control (SYSCTRL)** register selects which sleep mode is used (see page 155). For more information about the behavior of the sleep modes, see "System Control" on page 215.

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

2.7.1 Entering Sleep Modes

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

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

2.7.1.1 Wait for Interrupt

The wait for interrupt instruction, WFI, causes immediate entry to sleep mode unless the wake-up condition is true (see "Wake Up from WFI or Sleep-on-Exit" on page 104). When the processor executes a WFI instruction, it stops executing instructions and enters sleep mode. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information.

2.7.1.2 Wait for Event

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

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

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

2.7.1.3 Sleep-on-Exit

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

2.7.2 Wake Up from Sleep Mode

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

2.7.2.1 Wake Up from WFI or Sleep-on-Exit

Normally, the processor wakes up only when the NVIC detects an exception with sufficient priority to cause exception entry. Some embedded systems might have to execute system restore tasks after the processor wakes up and before executing an interrupt handler. Entry to the interrupt handler can be delayed by setting the PRIMASK bit and clearing the FAULTMASK bit. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor clears PRIMASK. For more information about **PRIMASK** and **FAULTMASK**, see page 74 and page 75.

2.7.2.2 Wake Up from WFE

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

In addition, if the SEVONPEND bit in the **SYSCTRL** register is set, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about **SYSCTRL**, see page 155.

2.8 Instruction Set Summary

The processor implements a version of the Thumb instruction set. Table 2-13 on page 104 lists the supported instructions.

Note: In Table 2-13 on page 104:

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

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

Table 2-13. Cortex-M4F Instruction Summary

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

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

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

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

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

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

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

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

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

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

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

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

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

3 Cortex-M4 Peripherals

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

- SysTick (see page 112)

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

- Nested Vectored Interrupt Controller (NVIC) (see page 113)
  - Facilitates low-latency exception and interrupt handling
  - Controls power management
  - Implements system control registers

- System Control Block (SCB) (see page 114)

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

- Memory Protection Unit (MPU) (see page 114)

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

- Floating-Point Unit (FPU) (see page 119)

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

Table 3-1 on page 111 shows the address map of the Private Peripheral Bus (PPB). Some peripheral register regions are split into two address regions, as indicated by two addresses listed.

Table 3-1. Core Peripheral Register Regions

Address	Core Peripheral	Description (see page ...)
0xE000.E010-0xE000.E01F	System Timer	112
0xE000.E100-0xE000.E4EF 0xE000.EF00-0xE000.EF03	Nested Vectored Interrupt Controller	113
0xE000.E008-0xE000.E00F 0xE000.ED00-0xE000.ED3F	System Control Block	114
0xE000.ED90-0xE000.EDB8	Memory Protection Unit	114
0xE000.EF30-0xE000.EF44	Floating Point Unit	119

3.1 Functional Description

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

3.1.1 System Timer (SysTick)

Cortex-M4 includes an integrated system timer, SysTick, which provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used in several different ways, for example as:

- An RTOS tick timer that fires at a programmable rate (for example, 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the system clock.
- A variable rate alarm or signal timer—the duration is range-dependent on the reference clock used and the dynamic range of the counter.
- A simple counter used to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The `COUNT` bit in the **STCTRL** control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

The timer consists of three registers:

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

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

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

The SysTick counter runs on either the system clock or the precision internal oscillator (PIOSC) divided by 4. If this clock signal is stopped for low power mode, the SysTick counter stops. SysTick can be kept running during Deep-sleep mode by setting the `CLK_SRC` bit in the **SysTick Control and Status Register (STCTRL)** register and ensuring that the `PIOSCPD` bit in the **Deep Sleep Clock Configuration (DSLPCLKCFG)** register is clear. Ensure software uses aligned word accesses to access the SysTick registers.

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

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

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

3.1.2 Nested Vectored Interrupt Controller (NVIC)

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

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

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

3.1.2.1 Level-Sensitive and Pulse Interrupts

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

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

When the processor enters the ISR, it automatically removes the pending state from the interrupt (see “Hardware and Software Control of Interrupts” on page 113 for more information). For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. As a result, the peripheral can hold the interrupt signal asserted until it no longer needs servicing.

3.1.2.2 Hardware and Software Control of Interrupts

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

- The NVIC detects that the interrupt signal is High and the interrupt is not active.
- The NVIC detects a rising edge on the interrupt signal.
- Software writes to the corresponding interrupt set-pending register bit, or to the **Software Trigger Interrupt (SWTRIG)** register to make a Software-Generated Interrupt pending. See the `INT` bit in the **PEND0** register on page 135 or **SWTRIG** on page 145.

A pending interrupt remains pending until one of the following:

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

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

3.1.3 System Control Block (SCB)

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

3.1.4 Memory Protection Unit (MPU)

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

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

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

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

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

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

Configuration of MPU regions is based on memory types (see “Memory Regions, Types and Attributes” on page 83 for more information).

Table 3-2 on page 115 shows the possible MPU region attributes. See the section called “MPU Configuration for a Tiva™ C Series Microcontroller” on page 119 for guidelines for programming a microcontroller implementation.

Table 3-2. Memory Attributes Summary

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

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

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

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

The processor does not support unaligned accesses to MPU registers.

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

3.1.4.1 Updating an MPU Region

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

Updating an MPU Region Using Separate Words

This example simple code configures one region:

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

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

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

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

Software must use memory barrier instructions:

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

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

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

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

Updating an MPU Region Using Multi-Word Writes

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

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

An `STM` instruction can be used to optimize this:

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

This operation can be done in two words for prepacked information, meaning that the **MPU Region Base Address (MPUBASE)** register (see page 179) contains the required region number and has the `VALID` bit set. This method can be used when the data is statically packed, for example in a boot loader:

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

Subregions

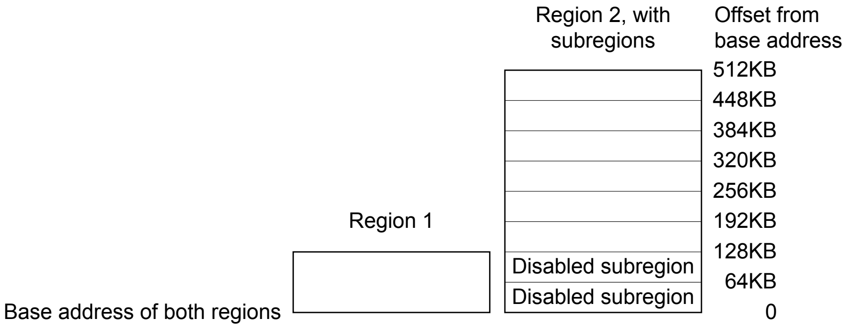
Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the SRD field of the **MPU Region Attribute and Size (MPUATTR)** register (see page 181) to disable a subregion. The least-significant bit of the SRD field controls the first subregion, and the most-significant bit controls the last subregion. Disabling a subregion means another region overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion, the MPU issues a fault.

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

Example of SRD Use

Two regions with the same base address overlap. Region one is 128 KB, and region two is 512 KB. To ensure the attributes from region one apply to the first 128 KB region, configure the SRD field for region two to 0x03 to disable the first two subregions, as Figure 3-1 on page 117 shows.

Figure 3-1. SRD Use Example



3.1.4.2 MPU Access Permission Attributes

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

Table 3-3 on page 117 shows the encodings for the TEX, C, B, and S access permission bits. All encodings are shown for completeness, however the current implementation of the Cortex-M4 does not support the concept of cacheability or shareability. Refer to the section called “MPU Configuration for a Tiva™ C Series Microcontroller” on page 119 for information on programming the MPU for TM4C1232C3PM implementations.

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

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

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

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

[a] The MPU ignores the value of this bit.

Table 3-4 on page 118 shows the cache policy for memory attribute encodings with a TEX value in the range of 0x4-0x7.

Table 3-4. Cache Policy for Memory Attribute Encoding

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

Table 3-5 on page 118 shows the AP encodings in the **MPUATTR** register that define the access permissions for privileged and unprivileged software.

Table 3-5. AP Bit Field Encoding

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

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

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

MPU Configuration for a Tiva™ C Series Microcontroller

Tiva™ C Series microcontrollers have only a single processor and no caches. As a result, the MPU should be programmed as shown in Table 3-6 on page 119.

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

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

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

3.1.4.3 MPU Mismatch

When an access violates the MPU permissions, the processor generates a memory management fault (see “Exceptions and Interrupts” on page 81 for more information). The **MFAULTSTAT** register indicates the cause of the fault. See page 166 for more information.

3.1.5 Floating-Point Unit (FPU)

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

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

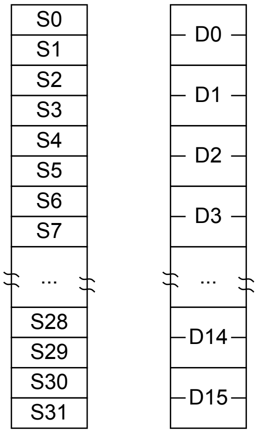
The Cortex-M4F FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions. The FPU provides floating-point computation functionality that is compliant with the ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic, referred to as the IEEE 754 standard. The FPU's single-precision extension registers can also be accessed as 16 doubleword registers for load, store, and move operations.

3.1.5.1 FPU Views of the Register Bank

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

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

Figure 3-2. FPU Register Bank



The mapping between the registers is as follows:

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

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

3.1.5.2 Modes of Operation

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

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

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

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

3.1.5.3 Compliance with the IEEE 754 standard

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

3.1.5.4 Complete Implementation of the IEEE 754 standard

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

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

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

3.1.5.5 IEEE 754 standard implementation choices

NaN handling

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

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

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

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

Table 3-7. QNaN and SNaN Handling

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

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

Comparisons

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

Underflow

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

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

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

3.1.5.6 Exceptions

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

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

3.1.5.7 Enabling the FPU

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

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

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

3.2 Register Map

Table 3-8 on page 123 lists the Cortex-M4 Peripheral SysTick, NVIC, MPU, FPU and SCB registers. The offset listed is a hexadecimal increment to the register's address, relative to the Core Peripherals base address of 0xE000.E000.

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

Table 3-8. Peripherals Register Map

Offset	Name	Type	Reset	Description	See page
System Timer (SysTick) Registers					
0x010	STCTRL	RW	0x0000.0004	SysTick Control and Status Register	127
0x014	STRELOAD	RW	-	SysTick Reload Value Register	129
0x018	STCURRENT	RWC	-	SysTick Current Value Register	130
Nested Vectored Interrupt Controller (NVIC) Registers					
0x100	EN0	RW	0x0000.0000	Interrupt 0-31 Set Enable	131
0x104	EN1	RW	0x0000.0000	Interrupt 32-63 Set Enable	131
0x108	EN2	RW	0x0000.0000	Interrupt 64-95 Set Enable	131
0x10C	EN3	RW	0x0000.0000	Interrupt 96-127 Set Enable	131
0x110	EN4	RW	0x0000.0000	Interrupt 128-138 Set Enable	132
0x180	DIS0	RW	0x0000.0000	Interrupt 0-31 Clear Enable	133
0x184	DIS1	RW	0x0000.0000	Interrupt 32-63 Clear Enable	133
0x188	DIS2	RW	0x0000.0000	Interrupt 64-95 Clear Enable	133
0x18C	DIS3	RW	0x0000.0000	Interrupt 96-127 Clear Enable	133
0x190	DIS4	RW	0x0000.0000	Interrupt 128-138 Clear Enable	134
0x200	PEND0	RW	0x0000.0000	Interrupt 0-31 Set Pending	135
0x204	PEND1	RW	0x0000.0000	Interrupt 32-63 Set Pending	135

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

Offset	Name	Type	Reset	Description	See page
0x208	PEND2	RW	0x0000.0000	Interrupt 64-95 Set Pending	135
0x20C	PEND3	RW	0x0000.0000	Interrupt 96-127 Set Pending	135
0x210	PEND4	RW	0x0000.0000	Interrupt 128-138 Set Pending	136
0x280	UNPEND0	RW	0x0000.0000	Interrupt 0-31 Clear Pending	137
0x284	UNPEND1	RW	0x0000.0000	Interrupt 32-63 Clear Pending	137
0x288	UNPEND2	RW	0x0000.0000	Interrupt 64-95 Clear Pending	137
0x28C	UNPEND3	RW	0x0000.0000	Interrupt 96-127 Clear Pending	137
0x290	UNPEND4	RW	0x0000.0000	Interrupt 128-138 Clear Pending	138
0x300	ACTIVE0	RO	0x0000.0000	Interrupt 0-31 Active Bit	139
0x304	ACTIVE1	RO	0x0000.0000	Interrupt 32-63 Active Bit	139
0x308	ACTIVE2	RO	0x0000.0000	Interrupt 64-95 Active Bit	139
0x30C	ACTIVE3	RO	0x0000.0000	Interrupt 96-127 Active Bit	139
0x310	ACTIVE4	RO	0x0000.0000	Interrupt 128-138 Active Bit	140
0x400	PRI0	RW	0x0000.0000	Interrupt 0-3 Priority	141
0x404	PRI1	RW	0x0000.0000	Interrupt 4-7 Priority	141
0x408	PRI2	RW	0x0000.0000	Interrupt 8-11 Priority	141
0x40C	PRI3	RW	0x0000.0000	Interrupt 12-15 Priority	141
0x410	PRI4	RW	0x0000.0000	Interrupt 16-19 Priority	141
0x414	PRI5	RW	0x0000.0000	Interrupt 20-23 Priority	141
0x418	PRI6	RW	0x0000.0000	Interrupt 24-27 Priority	141
0x41C	PRI7	RW	0x0000.0000	Interrupt 28-31 Priority	141
0x420	PRI8	RW	0x0000.0000	Interrupt 32-35 Priority	141
0x424	PRI9	RW	0x0000.0000	Interrupt 36-39 Priority	141
0x428	PRI10	RW	0x0000.0000	Interrupt 40-43 Priority	141
0x42C	PRI11	RW	0x0000.0000	Interrupt 44-47 Priority	141
0x430	PRI12	RW	0x0000.0000	Interrupt 48-51 Priority	141
0x434	PRI13	RW	0x0000.0000	Interrupt 52-55 Priority	141
0x438	PRI14	RW	0x0000.0000	Interrupt 56-59 Priority	141
0x43C	PRI15	RW	0x0000.0000	Interrupt 60-63 Priority	141
0x440	PRI16	RW	0x0000.0000	Interrupt 64-67 Priority	143
0x444	PRI17	RW	0x0000.0000	Interrupt 68-71 Priority	143
0x448	PRI18	RW	0x0000.0000	Interrupt 72-75 Priority	143

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

Offset	Name	Type	Reset	Description	See page
0x44C	PRI19	RW	0x0000.0000	Interrupt 76-79 Priority	143
0x450	PRI20	RW	0x0000.0000	Interrupt 80-83 Priority	143
0x454	PRI21	RW	0x0000.0000	Interrupt 84-87 Priority	143
0x458	PRI22	RW	0x0000.0000	Interrupt 88-91 Priority	143
0x45C	PRI23	RW	0x0000.0000	Interrupt 92-95 Priority	143
0x460	PRI24	RW	0x0000.0000	Interrupt 96-99 Priority	143
0x464	PRI25	RW	0x0000.0000	Interrupt 100-103 Priority	143
0x468	PRI26	RW	0x0000.0000	Interrupt 104-107 Priority	143
0x46C	PRI27	RW	0x0000.0000	Interrupt 108-111 Priority	143
0x470	PRI28	RW	0x0000.0000	Interrupt 112-115 Priority	143
0x474	PRI29	RW	0x0000.0000	Interrupt 116-119 Priority	143
0x478	PRI30	RW	0x0000.0000	Interrupt 120-123 Priority	143
0x47C	PRI31	RW	0x0000.0000	Interrupt 124-127 Priority	143
0x480	PRI32	RW	0x0000.0000	Interrupt 128-131 Priority	143
0x484	PRI33	RW	0x0000.0000	Interrupt 132-135 Priority	143
0x488	PRI34	RW	0x0000.0000	Interrupt 136-138 Priority	143
0xF00	SWTRIG	WO	0x0000.0000	Software Trigger Interrupt	145
System Control Block (SCB) Registers					
0x008	ACTLR	RW	0x0000.0000	Auxiliary Control	146
0xD00	CPUID	RO	0x410F.C241	CPU ID Base	148
0xD04	INTCTRL	RW	0x0000.0000	Interrupt Control and State	149
0xD08	VTABLE	RW	0x0000.0000	Vector Table Offset	152
0xD0C	APINT	RW	0xFA05.0000	Application Interrupt and Reset Control	153
0xD10	SYSCTRL	RW	0x0000.0000	System Control	155
0xD14	CFGCTRL	RW	0x0000.0200	Configuration and Control	157
0xD18	SYSPRI1	RW	0x0000.0000	System Handler Priority 1	159
0xD1C	SYSPRI2	RW	0x0000.0000	System Handler Priority 2	160
0xD20	SYSPRI3	RW	0x0000.0000	System Handler Priority 3	161
0xD24	SYSHNDCTRL	RW	0x0000.0000	System Handler Control and State	162
0xD28	FAULTSTAT	RW1C	0x0000.0000	Configurable Fault Status	166
0xD2C	HFAULTSTAT	RW1C	0x0000.0000	Hard Fault Status	172
0xD34	MMADDR	RW	-	Memory Management Fault Address	173

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

Offset	Name	Type	Reset	Description	See page
0xD38	FAULTADDR	RW	-	Bus Fault Address	174
Memory Protection Unit (MPU) Registers					
0xD90	MPUTYPE	RO	0x0000.0800	MPU Type	175
0xD94	MPUCTRL	RW	0x0000.0000	MPU Control	176
0xD98	MPUNUMBER	RW	0x0000.0000	MPU Region Number	178
0xD9C	MPUBASE	RW	0x0000.0000	MPU Region Base Address	179
0xDA0	MPUATTR	RW	0x0000.0000	MPU Region Attribute and Size	181
0xDA4	MPUBASE1	RW	0x0000.0000	MPU Region Base Address Alias 1	179
0xDA8	MPUATTR1	RW	0x0000.0000	MPU Region Attribute and Size Alias 1	181
0xDAC	MPUBASE2	RW	0x0000.0000	MPU Region Base Address Alias 2	179
0xDB0	MPUATTR2	RW	0x0000.0000	MPU Region Attribute and Size Alias 2	181
0xDB4	MPUBASE3	RW	0x0000.0000	MPU Region Base Address Alias 3	179
0xDB8	MPUATTR3	RW	0x0000.0000	MPU Region Attribute and Size Alias 3	181
Floating-Point Unit (FPU) Registers					
0xD88	CPAC	RW	0x0000.0000	Coprocessor Access Control	184
0xF34	FPCC	RW	0xC000.0000	Floating-Point Context Control	185
0xF38	FPCA	RW	-	Floating-Point Context Address	187
0xF3C	FPDSC	RW	0x0000.0000	Floating-Point Default Status Control	188

3.3 System Timer (SysTick) Register Descriptions

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

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

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

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

SysTick Control and Status Register (STCTRL)

Base 0xE000.E000
Offset 0x010
Type RW, reset 0x0000.0004

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													CLK_SRC	INTEN	ENABLE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	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	COUNT	RO	0	Count Flag Value Description 0 The SysTick timer has not counted to 0 since the last time this bit was read. 1 The SysTick timer has counted to 0 since the last time this bit was read. This bit is cleared by a read of the register or if the **STCURRENT** register is written with any value. If read by the debugger using the DAP, this bit is cleared only if the `MasterType` bit in the **AHB-AP Control Register** is clear. Otherwise, the `COUNT` bit is not changed by the debugger read. See the *ARM® Debug Interface V5 Architecture Specification* for more information on `MasterType`.
15:3	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	CLK_SRC	RW	1	Clock Source Value Description 0 Precision internal oscillator (PIOSC) divided by 4 1 System clock

Bit/Field	Name	Type	Reset	Description
1	INTEN	RW	0	Interrupt Enable Value Description 0 Interrupt generation is disabled. Software can use the COUNT bit to determine if the counter has ever reached 0. 1 An interrupt is generated to the NVIC when SysTick counts to 0.
0	ENABLE	RW	0	Enable Value Description 0 The counter is disabled. 1 Enables SysTick to operate in a multi-shot way. That is, the counter loads the RELOAD value and begins counting down. On reaching 0, the COUNT bit is set and an interrupt is generated if enabled by INTEN. The counter then loads the RELOAD value again and begins counting.

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

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

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

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

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

SysTick Reload Value Register (STRELOAD)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								RELOAD							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

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

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

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

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

SysTick Current Value Register (STCURRENT)

Base 0xE000.E000
Offset 0x018
Type RWC, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								CURRENT							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CURRENT															
Type	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC
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.
23:0	CURRENT	RWC	0x00.0000	Current Value This field contains the current value at the time the register is accessed. No read-modify-write protection is provided, so change with care. This register is write-clear. Writing to it with any value clears the register. Clearing this register also clears the `COUNT` bit of the **STCTRL** register.

3.4 NVIC Register Descriptions

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

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

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

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

Before programming the **VTABLE** register to relocate the vector table, ensure the vector table entries of the new vector table are set up for fault handlers, NMI, and all enabled exceptions such as interrupts. For more information, see page 152.

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

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

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

Register 7: Interrupt 96-127 Set Enable (EN3), offset 0x10C

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

The **ENn** registers enable interrupts and show which interrupts are enabled. Bit 0 of **EN0** corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of **EN1** corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of **EN2** corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of **EN3** corresponds to Interrupt 96; bit 31 corresponds to Interrupt 127. Bit 0 of **EN4** (see page 132) corresponds to Interrupt 128; bit 10 corresponds to Interrupt 138.

See Table 2-9 on page 93 for interrupt assignments.

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

Interrupt 0-31 Set Enable (EN0)

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

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

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

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

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

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

Register 8: Interrupt 128-138 Set Enable (EN4), offset 0x110

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

The **EN4** register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 128; bit 10 corresponds to Interrupt 138. See Table 2-9 on page 93 for interrupt assignments.

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

Interrupt 128-138 Set Enable (EN4)

Base 0xE000.E000
Offset 0x110
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:11	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.
10:0	INT	RW	0x0	Interrupt Enable

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

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **DIS4** register.

Register 9: Interrupt 0-31 Clear Enable (DIS0), offset 0x180

Register 10: Interrupt 32-63 Clear Enable (DIS1), offset 0x184

Register 11: Interrupt 64-95 Clear Enable (DIS2), offset 0x188

Register 12: Interrupt 96-127 Clear Enable (DIS3), offset 0x18C

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

The **DISn** registers disable interrupts. Bit 0 of **DIS0** corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of **DIS1** corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of **DIS2** corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of **DIS3** corresponds to Interrupt 96; bit 31 corresponds to Interrupt 127. Bit 0 of **DIS4** (see page 134) corresponds to Interrupt 128; bit 10 corresponds to Interrupt 138.

See Table 2-9 on page 93 for interrupt assignments.

Interrupt 0-31 Clear Enable (DIS0)

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

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

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

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

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

Register 13: Interrupt 128-138 Clear Enable (DIS4), offset 0x190

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

The **DIS4** register disables interrupts. Bit 0 corresponds to Interrupt 128; bit 10 corresponds to Interrupt 138. See Table 2-9 on page 93 for interrupt assignments.

Interrupt 128-138 Clear Enable (DIS4)

Base 0xE000.E000
Offset 0x190
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:11	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.
10:0	INT	RW	0x0	Interrupt Disable Value Description 0 On a read, indicates the interrupt is disabled. On a write, no effect. 1 On a read, indicates the interrupt is enabled. On a write, clears the corresponding `INT[n]` bit in the **EN4** register, disabling interrupt [n].

Register 14: Interrupt 0-31 Set Pending (PEND0), offset 0x200

Register 15: Interrupt 32-63 Set Pending (PEND1), offset 0x204

Register 16: Interrupt 64-95 Set Pending (PEND2), offset 0x208

Register 17: Interrupt 96-127 Set Pending (PEND3), offset 0x20C

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

The **PENDn** registers force interrupts into the pending state and show which interrupts are pending. Bit 0 of **PEND0** corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of **PEND1** corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of **PEND2** corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of **PEND3** corresponds to Interrupt 96; bit 31 corresponds to Interrupt 127. Bit 0 of **PEND4** (see page 136) corresponds to Interrupt 128; bit 10 corresponds to Interrupt 138.

See Table 2-9 on page 93 for interrupt assignments.

Interrupt 0-31 Set Pending (PEND0)

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

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

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

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

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

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

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

Register 18: Interrupt 128-138 Set Pending (PEND4), offset 0x210

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

The **PEND4** register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 128; bit 10 corresponds to Interrupt 138. See Table 2-9 on page 93 for interrupt assignments.

Interrupt 128-138 Set Pending (PEND4)

Base 0xE000.E000
Offset 0x210
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:11	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.
10:0	INT	RW	0x0	Interrupt Set Pending

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

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

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **UNPEND4** register.

Register 19: Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280

Register 20: Interrupt 32-63 Clear Pending (UNPEND1), offset 0x284

Register 21: Interrupt 64-95 Clear Pending (UNPEND2), offset 0x288

Register 22: Interrupt 96-127 Clear Pending (UNPEND3), offset 0x28C

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

The **UNPENDn** registers show which interrupts are pending and remove the pending state from interrupts. Bit 0 of **UNPEND0** corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of **UNPEND1** corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of **UNPEND2** corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of **UNPEND3** corresponds to Interrupt 96; bit 31 corresponds to Interrupt 127. Bit 0 of **UNPEND4** (see page 138) corresponds to Interrupt 128; bit 10 corresponds to Interrupt 138.

See Table 2-9 on page 93 for interrupt assignments.

Interrupt 0-31 Clear Pending (UNPEND0)

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

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

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

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

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

Register 23: Interrupt 128-138 Clear Pending (UNPEND4), offset 0x290

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

The **UNPEND4** register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 128; bit 10 corresponds to Interrupt 138. See Table 2-9 on page 93 for interrupt assignments.

Interrupt 128-138 Clear Pending (UNPEND4)

Base 0xE000.E000
Offset 0x290
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:11	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.
10:0	INT	RW	0x0	Interrupt Clear Pending

Value	Description
0	On a read, indicates that the interrupt is not pending. On a write, no effect.
1	On a read, indicates that the interrupt is pending. On a write, clears the corresponding `INT[n]` bit in the **PEND4** register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt.

Register 24: Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300

Register 25: Interrupt 32-63 Active Bit (ACTIVE1), offset 0x304

Register 26: Interrupt 64-95 Active Bit (ACTIVE2), offset 0x308

Register 27: Interrupt 96-127 Active Bit (ACTIVE3), offset 0x30C

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

The **UNPENDn** registers indicate which interrupts are active. Bit 0 of **ACTIVE0** corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of **ACTIVE1** corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of **ACTIVE2** corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of **ACTIVE3** corresponds to Interrupt 96; bit 31 corresponds to Interrupt 127. Bit 0 of **ACTIVE4** (see page 140) corresponds to Interrupt 128; bit 10 corresponds to Interrupt 138.

See Table 2-9 on page 93 for interrupt assignments.

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

Interrupt 0-31 Active Bit (ACTIVE0)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INT															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	INT	RO	0x0000.0000	Interrupt Active Value Description 0 The corresponding interrupt is not active. 1 The corresponding interrupt is active, or active and pending.

Register 28: Interrupt 128-138 Active Bit (ACTIVE4), offset 0x310

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

The **ACTIVE4** register indicates which interrupts are active. Bit 0 corresponds to Interrupt 128; bit 10 corresponds to Interrupt 131. See Table 2-9 on page 93 for interrupt assignments.

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

Interrupt 128-138 Active Bit (ACTIVE4)

Base 0xE000.E000
Offset 0x310
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					INT										
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:11	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.
10:0	INT	RO	0x0	Interrupt Active

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

Register 29: Interrupt 0-3 Priority (PRI0), offset 0x400

Register 30: Interrupt 4-7 Priority (PRI1), offset 0x404

Register 31: Interrupt 8-11 Priority (PRI2), offset 0x408

Register 32: Interrupt 12-15 Priority (PRI3), offset 0x40C

Register 33: Interrupt 16-19 Priority (PRI4), offset 0x410

Register 34: Interrupt 20-23 Priority (PRI5), offset 0x414

Register 35: Interrupt 24-27 Priority (PRI6), offset 0x418

Register 36: Interrupt 28-31 Priority (PRI7), offset 0x41C

Register 37: Interrupt 32-35 Priority (PRI8), offset 0x420

Register 38: Interrupt 36-39 Priority (PRI9), offset 0x424

Register 39: Interrupt 40-43 Priority (PRI10), offset 0x428

Register 40: Interrupt 44-47 Priority (PRI11), offset 0x42C

Register 41: Interrupt 48-51 Priority (PRI12), offset 0x430

Register 42: Interrupt 52-55 Priority (PRI13), offset 0x434

Register 43: Interrupt 56-59 Priority (PRI14), offset 0x438

Register 44: Interrupt 60-63 Priority (PRI15), offset 0x43C

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

The **PRIn** registers (see also page 143) provide 3-bit priority fields for each interrupt. These registers are byte accessible. Each register holds four priority fields that are assigned to interrupts as follows:

PRIn Register Bit Field	Interrupt
Bits 31:29	Interrupt [4n+3]
Bits 23:21	Interrupt [4n+2]
Bits 15:13	Interrupt [4n+1]
Bits 7:5	Interrupt [4n]

See Table 2-9 on page 93 for interrupt assignments.

Each priority level can be split into separate group priority and subpriority fields. The PRIGROUP field in the **Application Interrupt and Reset Control (APINT)** register (see page 153) indicates the position of the binary point that splits the priority and subpriority fields.

These registers can only be accessed from privileged mode.

Interrupt 0-3 Priority (PRI0)

Base 0xE000.E000
Offset 0x400
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INTD			reserved					INTC			reserved				
Type	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTB			reserved					INTA			reserved				
Type	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	INTD	RW	0x0	Interrupt Priority for Interrupt [4n+3] This field holds a priority value, 0-7, for the interrupt with the number [4n+3], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
28:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	INTC	RW	0x0	Interrupt Priority for Interrupt [4n+2] This field holds a priority value, 0-7, for the interrupt with the number [4n+2], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
20:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:13	INTB	RW	0x0	Interrupt Priority for Interrupt [4n+1] This field holds a priority value, 0-7, for the interrupt with the number [4n+1], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
12:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	INTA	RW	0x0	Interrupt Priority for Interrupt [4n] This field holds a priority value, 0-7, for the interrupt with the number [4n], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 45: Interrupt 64-67 Priority (PRI16), offset 0x440

Register 46: Interrupt 68-71 Priority (PRI17), offset 0x444

Register 47: Interrupt 72-75 Priority (PRI18), offset 0x448

Register 48: Interrupt 76-79 Priority (PRI19), offset 0x44C

Register 49: Interrupt 80-83 Priority (PRI20), offset 0x450

Register 50: Interrupt 84-87 Priority (PRI21), offset 0x454

Register 51: Interrupt 88-91 Priority (PRI22), offset 0x458

Register 52: Interrupt 92-95 Priority (PRI23), offset 0x45C

Register 53: Interrupt 96-99 Priority (PRI24), offset 0x460

Register 54: Interrupt 100-103 Priority (PRI25), offset 0x464

Register 55: Interrupt 104-107 Priority (PRI26), offset 0x468

Register 56: Interrupt 108-111 Priority (PRI27), offset 0x46C

Register 57: Interrupt 112-115 Priority (PRI28), offset 0x470

Register 58: Interrupt 116-119 Priority (PRI29), offset 0x474

Register 59: Interrupt 120-123 Priority (PRI30), offset 0x478

Register 60: Interrupt 124-127 Priority (PRI31), offset 0x47C

Register 61: Interrupt 128-131 Priority (PRI32), offset 0x480

Register 62: Interrupt 132-135 Priority (PRI33), offset 0x484

Register 63: Interrupt 136-138 Priority (PRI34), offset 0x488

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

The **PRIn** registers (see also page 141) provide 3-bit priority fields for each interrupt. These registers are byte accessible. Each register holds four priority fields that are assigned to interrupts as follows:

PRIn Register Bit Field	Interrupt
Bits 31:29	Interrupt [4n+3]
Bits 23:21	Interrupt [4n+2]
Bits 15:13	Interrupt [4n+1]
Bits 7:5	Interrupt [4n]

See Table 2-9 on page 93 for interrupt assignments.

Each priority level can be split into separate group priority and subpriority fields. The `PRIGROUP` field in the **Application Interrupt and Reset Control (APINT)** register (see page 153) indicates the position of the binary point that splits the priority and subpriority fields .

These registers can only be accessed from privileged mode.

Interrupt 64-67 Priority (PRI16)

Base 0xE000.E000
Offset 0x440
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INTD			reserved					INTC			reserved				
Type	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTB			reserved					INTA			reserved				
Type	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	INTD	RW	0x0	Interrupt Priority for Interrupt [4n+3] This field holds a priority value, 0-7, for the interrupt with the number [4n+3], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
28:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	INTC	RW	0x0	Interrupt Priority for Interrupt [4n+2] This field holds a priority value, 0-7, for the interrupt with the number [4n+2], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
20:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:13	INTB	RW	0x0	Interrupt Priority for Interrupt [4n+1] This field holds a priority value, 0-7, for the interrupt with the number [4n+1], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
12:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	INTA	RW	0x0	Interrupt Priority for Interrupt [4n] This field holds a priority value, 0-7, for the interrupt with the number [4n], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 64: Software Trigger Interrupt (SWTRIG), offset 0xF00

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

Writing an interrupt number to the **SWTRIG** register generates a Software Generated Interrupt (SGI). See Table 2-9 on page 93 for interrupt assignments.

When the `MAINPEND` bit in the **Configuration and Control (CFGCTRL)** register (see page 157) is set, unprivileged software can access the **SWTRIG** register.

Software Trigger Interrupt (SWTRIG)

Base 0xE000.E000
Offset 0xF00
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								INTID							
Type	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	INTID	WO	0x00	Interrupt ID This field holds the interrupt ID of the required SGI. For example, a value of 0x3 generates an interrupt on IRQ3.

3.5 System Control Block (SCB) Register Descriptions

This section lists and describes the System Control Block (SCB) registers, in numerical order by address offset. The SCB registers can only be accessed from privileged mode.

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

Register 65: Auxiliary Control (ACTLR), offset 0x008

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

The **ACTLR** register provides disable bits for `IT` folding, write buffer use for accesses to the default memory map, and interruption of multi-cycle instructions. By default, this register is set to provide optimum performance from the Cortex-M4 processor and does not normally require modification.

Auxiliary Control (ACTLR)

Base 0xE000.E000
Offset 0x008
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						DISOOFP	DISFPCA	reserved					DISFOLD	DISWBUF	DISMCYC
Type	RO	RO	RO	RO	RO	RO	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	DISOOFP	RW	0	Disable Out-Of-Order Floating Point Disables floating-point instructions completing out of order with respect to integer instructions.
8	DISFPCA	RW	0	Disable CONTROL.FPCA Disable automatic update of the `FPCA` bit in the **CONTROL** register. **Important:** Two bits control when `FPCA` can be enabled: the `ASPEN` bit in the **Floating-Point Context Control (FPCC)** register and the `DISFPCA` bit in the **Auxiliary Control (ACTLR)** register.
7:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DISFOLD	RW	0	Disable IT Folding Value Description 0 No effect. 1 Disables `IT` folding. In some situations, the processor can start executing the first instruction in an `IT` block while it is still executing the `IT` instruction. This behavior is called *IT folding*, and improves performance, However, `IT` folding can cause jitter in looping. If a task must avoid jitter, set the `DISFOLD` bit before executing the task, to disable `IT` folding.

Bit/Field	Name	Type	Reset	Description
1	DISWBUF	RW	0	Disable Write Buffer Value Description 0 No effect. 1 Disables write buffer use during default memory map accesses. In this situation, all bus faults are precise bus faults but performance is decreased because any store to memory must complete before the processor can execute the next instruction. **Note:** This bit only affects write buffers implemented in the Cortex-M4 processor.
0	DISMCYC	RW	0	Disable Interrupts of Multiple Cycle Instructions Value Description 0 No effect. 1 Disables interruption of load multiple and store multiple instructions. In this situation, the interrupt latency of the processor is increased because any `LDM` or `STM` must complete before the processor can stack the current state and enter the interrupt handler.

Register 66: CPU ID Base (CPUID), offset 0xD00

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

The **CPUID** register contains the ARM® Cortex™-M4 processor part number, version, and implementation information.

CPU ID Base (CPUID)

Base 0xE000.E000
Offset 0xD00
Type RO, reset 0x410F.C241

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IMP								VAR				CON			
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	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
	PARTNO												REV			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:24	IMP	RO	0x41	Implementer Code Value Description 0x41 ARM
23:20	VAR	RO	0x0	Variant Number Value Description 0x0 The rn value in the rnpn product revision identifier, for example, the 0 in r0p0.
19:16	CON	RO	0xF	Constant Value Description 0xF Always reads as 0xF.
15:4	PARTNO	RO	0xC24	Part Number Value Description 0xC24 Cortex-M4 processor.
3:0	REV	RO	0x1	Revision Number Value Description 0x1 The pn value in the rnpn product revision identifier, for example, the 1 in r0p1.

Register 67: Interrupt Control and State (INTCTRL), offset 0xD04

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

The **INCTRL** register provides a set-pending bit for the NMI exception, and set-pending and clear-pending bits for the PendSV and SysTick exceptions. In addition, bits in this register indicate the exception number of the exception being processed, whether there are preempted active exceptions, the exception number of the highest priority pending exception, and whether any interrupts are pending.

When writing to **INCTRL**, the effect is unpredictable when writing a 1 to both the PENDSV and UNPENDSV bits, or writing a 1 to both the PENDSTSET and PENDSTCLR bits.

Interrupt Control and State (INTCTRL)

Base 0xE000.E000
Offset 0xD04
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NMISET	reserved		PENDSV	UNPENDSV	PENDSTSET	PENDSTCLR	reserved	ISRPRE	ISRPEND	reserved		VECPEND			
Type	RW	RO	RO	RW	WO	RW	WO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VECPEND				RETBASE	reserved			VECACT							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	NMISET	RW	0	NMI Set Pending

Value	Description
0	On a read, indicates an NMI exception is not pending. On a write, no effect.
1	On a read, indicates an NMI exception is pending. On a write, changes the NMI exception state to pending.

Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it registers the setting of this bit, and clears this bit on entering the interrupt handler. A read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.

Bit/Field	Name	Type	Reset	Description
30:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	PENDSV	RW	0	PendSV Set Pending

Value	Description
0	On a read, indicates a PendSV exception is not pending. On a write, no effect.
1	On a read, indicates a PendSV exception is pending. On a write, changes the PendSV exception state to pending.

Setting this bit is the only way to set the PendSV exception state to pending. This bit is cleared by writing a 1 to the UNPENDSV bit.

Bit/Field	Name	Type	Reset	Description
27	UNPENDSV	WO	0	PendSV Clear Pending Value Description 0 On a write, no effect. 1 On a write, removes the pending state from the PendSV exception. This bit is write only; on a register read, its value is unknown.
26	PENDSTSET	RW	0	SysTick Set Pending Value Description 0 On a read, indicates a SysTick exception is not pending. On a write, no effect. 1 On a read, indicates a SysTick exception is pending. On a write, changes the SysTick exception state to pending. This bit is cleared by writing a 1 to the `PENDSTCLR` bit.
25	PENDSTCLR	WO	0	SysTick Clear Pending Value Description 0 On a write, no effect. 1 On a write, removes the pending state from the SysTick exception. This bit is write only; on a register read, its value is unknown.
24	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	ISRPRE	RO	0	Debug Interrupt Handling Value Description 0 The release from halt does not take an interrupt. 1 The release from halt takes an interrupt. This bit is only meaningful in Debug mode and reads as zero when the processor is not in Debug mode.
22	ISRPEND	RO	0	Interrupt Pending Value Description 0 No interrupt is pending. 1 An interrupt is pending. This bit provides status for all interrupts excluding NMI and Faults.
21:20	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
19:12	VECPEND	RO	0x00	Interrupt Pending Vector Number This field contains the exception number of the highest priority pending enabled exception. The value indicated by this field includes the effect of the **BASEPRI** and **FAULTMASK** registers, but not any effect of the **PRIMASK** register. Value — Description 0x00 — No exceptions are pending 0x01 — Reserved 0x02 — NMI 0x03 — Hard fault 0x04 — Memory management fault 0x05 — Bus fault 0x06 — Usage fault 0x07-0x0A — Reserved 0x0B — SVCall 0x0C — Reserved for Debug 0x0D — Reserved 0x0E — PendSV 0x0F — SysTick 0x10 — Interrupt Vector 0 0x11 — Interrupt Vector 1 ... — ... 0x9A — Interrupt Vector 138
11	RETBASE	RO	0	Return to Base Value — Description 0 — There are preempted active exceptions to execute. 1 — There are no active exceptions, or the currently executing exception is the only active exception. This bit provides status for all interrupts excluding NMI and Faults. This bit only has meaning if the processor is currently executing an ISR (the **Interrupt Program Status (IPSR)** register is non-zero).
10:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	VECACT	RO	0x00	Interrupt Pending Vector Number This field contains the active exception number. The exception numbers can be found in the description for the `VECPEND` field. If this field is clear, the processor is in Thread mode. This field contains the same value as the `ISRNUM` field in the **IPSR** register. Subtract 16 from this value to obtain the IRQ number required to index into the **Interrupt Set Enable (ENn)**, **Interrupt Clear Enable (DISn)**, **Interrupt Set Pending (PENDn)**, **Interrupt Clear Pending (UNPENDn)**, and **Interrupt Priority (PRIn)** registers (see page 70).

Register 68: Vector Table Offset (VTABLE), offset 0xD08

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

The **VTABLE** register indicates the offset of the vector table base address from memory address 0x0000.0000.

Vector Table Offset (VTABLE)

Base 0xE000.E000
Offset 0xD08
Type RW, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OFFSET						reserved									
Type	RW	RW	RW	RW	RW	RW	RO	RO	RO	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	OFFSET	RW	0x000.00	Vector Table Offset When configuring the OFFSET field, the offset must be aligned to the number of exception entries in the vector table. Because there are 138 interrupts, the offset must be aligned on a 1024-byte boundary.
9:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 69: Application Interrupt and Reset Control (APINT), offset 0xD0C

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

The **APINT** register provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system. To write to this register, 0x05FA must be written to the VECTKEY field, otherwise the write is ignored.

The PRIGROUP field indicates the position of the binary point that splits the INTx fields in the **Interrupt Priority (PRIx)** registers into separate group priority and subpriority fields. Table 3-9 on page 153 shows how the PRIGROUP value controls this split. The bit numbers in the Group Priority Field and Subpriority Field columns in the table refer to the bits in the INTA field. For the INTB field, the corresponding bits are 15:13; for INTC, 23:21; and for INTD, 31:29.

Note: Determining preemption of an exception uses only the group priority field.

Table 3-9. Interrupt Priority Levels

PRIGROUP Bit Field	Binary Point[a]	Group Priority Field	Subpriority Field	Group Priorities	Subpriorities
0x0 - 0x4	bxxx.	[7:5]	None	8	1
0x5	bxx.y	[7:6]	[5]	4	2
0x6	bx.yy	[7]	[6:5]	2	4
0x7	b.yyy	None	[7:5]	1	8

[a] INTx field showing the binary point. An x denotes a group priority field bit, and a y denotes a subpriority field bit.

Application Interrupt and Reset Control (APINT)

Base 0xE000.E000
Offset 0xD0C
Type RW, reset 0xFA05.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:16	VECTKEY	RW	0xFA05	Register Key This field is used to guard against accidental writes to this register. 0x05FA must be written to this field in order to change the bits in this register. On a read, 0xFA05 is returned.
15	ENDIANESS	RO	0	Data Endianess The Tiva™ C Series implementation uses only little-endian mode so this is cleared to 0.
14:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
10:8	PRIGROUP	RW	0x0	Interrupt Priority Grouping This field determines the split of group priority from subpriority (see Table 3-9 on page 153 for more information).
7:3	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	SYSRESREQ	WO	0	System Reset Request Value Description 0 No effect. 1 Resets the core and all on-chip peripherals except the Debug interface. This bit is automatically cleared during the reset of the core and reads as 0.
1	VECTCLRACT	WO	0	Clear Active NMI / Fault This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.
0	VECTRESET	WO	0	System Reset This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.

Register 70: System Control (SYSCTRL), offset 0xD10

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

The **SYSCTRL** register controls features of entry to and exit from low-power state.

System Control (SYSCTRL)

Base 0xE000.E000
Offset 0xD10
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											SEVONPEND	reserved	SLEEPDEEP	SLEEPEXIT	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RO	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SEVONPEND	RW	0	Wake Up on Pending Value Description 0 Only enabled interrupts or events can wake up the processor; disabled interrupts are excluded. 1 Enabled events and all interrupts, including disabled interrupts, can wake up the processor. When an event or interrupt enters the pending state, the event signal wakes up the processor from `WFE`. If the processor is not waiting for an event, the event is registered and affects the next `WFE`. The processor also wakes up on execution of a `SEV` instruction or an external event.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	SLEEPDEEP	RW	0	Deep Sleep Enable Value Description 0 Use Sleep mode as the low power mode. 1 Use Deep-sleep mode as the low power mode.

Bit/Field	Name	Type	Reset	Description
1	SLEEPEXIT	RW	0	Sleep on ISR Exit Value Description 0 When returning from Handler mode to Thread mode, do not sleep when returning to Thread mode. 1 When returning from Handler mode to Thread mode, enter sleep or deep sleep on return from an ISR. Setting this bit enables an interrupt-driven application to avoid returning to an empty main application.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 71: Configuration and Control (CFGCTRL), offset 0xD14

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

The **CFGCTRL** register controls entry to Thread mode and enables: the handlers for NMI, hard fault and faults escalated by the **FAULTMASK** register to ignore bus faults; trapping of divide by zero and unaligned accesses; and access to the **SWTRIG** register by unprivileged software (see page 145).

Configuration and Control (CFGCTRL)

Base 0xE000.E000
Offset 0xD14
Type RW, reset 0x0000.0200

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						STKALIGN	BFHFNMIGN	reserved			DIV0	UNALIGNED	reserved	MAINPEND	BASETHR
Type	RO	RO	RO	RO	RO	RO	RW	RW	RO	RO	RO	RW	RW	RO	RW	RW
Reset	0	0	0	0	0	0	1	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	STKALIGN	RW	1	Stack Alignment on Exception Entry Value Description 0 The stack is 4-byte aligned. 1 The stack is 8-byte aligned. On exception entry, the processor uses bit 9 of the stacked **PSR** to indicate the stack alignment. On return from the exception, it uses this stacked bit to restore the correct stack alignment.
8	BFHFNMIGN	RW	0	Ignore Bus Fault in NMI and Fault This bit enables handlers with priority -1 or -2 to ignore data bus faults caused by load and store instructions. The setting of this bit applies to the hard fault, NMI, and **FAULTMASK** escalated handlers. Value Description 0 Data bus faults caused by load and store instructions cause a lock-up. 1 Handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions. Set this bit only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect control path problems and fix them.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
4	DIV0	RW	0	Trap on Divide by 0 This bit enables faulting or halting when the processor executes an `SDIV` or `UDIV` instruction with a divisor of 0. Value Description 0 Do not trap on divide by 0. A divide by zero returns a quotient of 0. 1 Trap on divide by 0.
3	UNALIGNED	RW	0	Trap on Unaligned Access Value Description 0 Do not trap on unaligned halfword and word accesses. 1 Trap on unaligned halfword and word accesses. An unaligned access generates a usage fault. Unaligned `LDM`, `STM`, `LDRD`, and `STRD` instructions always fault regardless of whether `UNALIGNED` is set.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	MAINPEND	RW	0	Allow Main Interrupt Trigger Value Description 0 Disables unprivileged software access to the **SWTRIG** register. 1 Enables unprivileged software access to the **SWTRIG** register (see page 145).
0	BASETHR	RW	0	Thread State Control Value Description 0 The processor can enter Thread mode only when no exception is active. 1 The processor can enter Thread mode from any level under the control of an EXC_RETURN value (see "Exception Return" on page 99 for more information).

Register 72: System Handler Priority 1 (SYSPRI1), offset 0xD18

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

The **SYSPRI1** register configures the priority level, 0 to 7 of the usage fault, bus fault, and memory management fault exception handlers. This register is byte-accessible.

System Handler Priority 1 (SYSPRI1)

Base 0xE000.E000
Offset 0xD18
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								USAGE			reserved				
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BUS			reserved					MEM			reserved				
Type	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	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
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	USAGE	RW	0x0	Usage Fault Priority This field configures the priority level of the usage fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
20:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:13	BUS	RW	0x0	Bus Fault Priority This field configures the priority level of the bus fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
12:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	MEM	RW	0x0	Memory Management Fault Priority This field configures the priority level of the memory management fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 73: System Handler Priority 2 (SYSPRI2), offset 0xD1C

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

The **SYSPRI2** register configures the priority level, 0 to 7 of the SVCall handler. This register is byte-accessible.

System Handler Priority 2 (SYSPRI2)

Base 0xE000.E000
Offset 0xD1C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SVC			reserved												
Type	RW	RW	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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
31:29	SVC	RW	0x0	SVCall Priority This field configures the priority level of SVCall. Configurable priority values are in the range 0-7, with lower values having higher priority.
28:0	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 74: System Handler Priority 3 (SYSPRI3), offset 0xD20

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

The **SYSPRI3** register configures the priority level, 0 to 7 of the SysTick exception and PendSV handlers. This register is byte-accessible.

System Handler Priority 3 (SYSPRI3)

Base 0xE000.E000
Offset 0xD20
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TICK			reserved					PENDSV			reserved				
Type	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DEBUG			reserved				
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	TICK	RW	0x0	SysTick Exception Priority This field configures the priority level of the SysTick exception. Configurable priority values are in the range 0-7, with lower values having higher priority.
28:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	PENDSV	RW	0x0	PendSV Priority This field configures the priority level of PendSV. Configurable priority values are in the range 0-7, with lower values having higher priority.
20:8	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	DEBUG	RW	0x0	Debug Priority This field configures the priority level of Debug. Configurable priority values are in the range 0-7, with lower values having higher priority.
4:0	reserved	RO	0x0.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 75: System Handler Control and State (SYSHNDCTRL), offset 0xD24

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

The **SYSHNDCTRL** register enables the system handlers, and indicates the pending status of the usage fault, bus fault, memory management fault, and SVC exceptions as well as the active status of the system handlers.

If a system handler is disabled and the corresponding fault occurs, the processor treats the fault as a hard fault.

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

Caution – Software that changes the value of an active bit in this register without correct adjustment to the stacked content can cause the processor to generate a fault exception. Ensure software that writes to this register retains and subsequently restores the current active status.

If the value of a bit in this register must be modified after enabling the system handlers, a read-modify-write procedure must be used to ensure that only the required bit is modified.

System Handler Control and State (SYSHNDCTRL)

Base 0xE000.E000
Offset 0xD24
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved													USAGE	BUS	MEM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SVC	BUSP	MEMP	USAGEP	TICK	PNDSV	reserved	MON	SVCA	reserved			USGA	reserved	BUSA	MEMA
Type	RW	RW	RW	RW	RW	RW	RO	RW	RW	RO	RO	RO	RW	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:19	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	USAGE	RW	0	Usage Fault Enable Value Description 0 Disables the usage fault exception. 1 Enables the usage fault exception.
17	BUS	RW	0	Bus Fault Enable Value Description 0 Disables the bus fault exception. 1 Enables the bus fault exception.

Bit/Field	Name	Type	Reset	Description
16	MEM	RW	0	Memory Management Fault Enable Value Description 0 Disables the memory management fault exception. 1 Enables the memory management fault exception.
15	SVC	RW	0	SVC Call Pending Value Description 0 An SVC call exception is not pending. 1 An SVC call exception is pending. This bit can be modified to change the pending status of the SVC call exception.
14	BUSP	RW	0	Bus Fault Pending Value Description 0 A bus fault exception is not pending. 1 A bus fault exception is pending. This bit can be modified to change the pending status of the bus fault exception.
13	MEMP	RW	0	Memory Management Fault Pending Value Description 0 A memory management fault exception is not pending. 1 A memory management fault exception is pending. This bit can be modified to change the pending status of the memory management fault exception.
12	USAGEP	RW	0	Usage Fault Pending Value Description 0 A usage fault exception is not pending. 1 A usage fault exception is pending. This bit can be modified to change the pending status of the usage fault exception.
11	TICK	RW	0	SysTick Exception Active Value Description 0 A SysTick exception is not active. 1 A SysTick exception is active. This bit can be modified to change the active status of the SysTick exception, however, see the Caution above before setting this bit.

Bit/Field	Name	Type	Reset	Description
10	PNDSV	RW	0	PendSV Exception Active Value Description 0 A PendSV exception is not active. 1 A PendSV exception is active. This bit can be modified to change the active status of the PendSV exception, however, see the Caution above before setting this bit.
9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MON	RW	0	Debug Monitor Active Value Description 0 The Debug monitor is not active. 1 The Debug monitor is active.
7	SVCA	RW	0	SVC Call Active Value Description 0 SVC call is not active. 1 SVC call is active. This bit can be modified to change the active status of the SVC call exception, however, see the Caution above before setting this bit.
6:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	USGA	RW	0	Usage Fault Active Value Description 0 Usage fault is not active. 1 Usage fault is active. This bit can be modified to change the active status of the usage fault exception, however, see the Caution above before setting this bit.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BUSA	RW	0	Bus Fault Active Value Description 0 Bus fault is not active. 1 Bus fault is active. This bit can be modified to change the active status of the bus fault exception, however, see the Caution above before setting this bit.

Bit/Field	Name	Type	Reset	Description
0	MEMA	RW	0	Memory Management Fault Active Value Description 0 Memory management fault is not active. 1 Memory management fault is active. This bit can be modified to change the active status of the memory management fault exception, however, see the Caution above before setting this bit.

Register 76: Configurable Fault Status (FAULTSTAT), offset 0xD28

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

The **FAULTSTAT** register indicates the cause of a memory management fault, bus fault, or usage fault. Each of these functions is assigned to a subregister as follows:

- **Usage Fault Status (UFAULTSTAT)**, bits 31:16
- **Bus Fault Status (BFAULTSTAT)**, bits 15:8
- **Memory Management Fault Status (MFAULTSTAT)**, bits 7:0

FAULTSTAT is byte accessible. **FAULTSTAT** or its subregisters can be accessed as follows:

- The complete **FAULTSTAT** register, with a word access to offset 0xD28
- The **MFAULTSTAT**, with a byte access to offset 0xD28
- The **MFAULTSTAT** and **BFAULTSTAT**, with a halfword access to offset 0xD28
- The **BFAULTSTAT**, with a byte access to offset 0xD29
- The **UFAULTSTAT**, with a halfword access to offset 0xD2A

Bits are cleared by writing a 1 to them.

In a fault handler, the true faulting address can be determined by:

1. Read and save the **Memory Management Fault Address (MMADDR)** or **Bus Fault Address (FAULTADDR)** value.

2. Read the `MMARV` bit in **MFAULTSTAT**, or the `BFARV` bit in **BFAULTSTAT** to determine if the **MMADDR** or **FAULTADDR** contents are valid.

Software must follow this sequence because another higher priority exception might change the **MMADDR** or **FAULTADDR** value. For example, if a higher priority handler preempts the current fault handler, the other fault might change the **MMADDR** or **FAULTADDR** value.

Configurable Fault Status (FAULTSTAT)

Base 0xE000.E000
Offset 0xD28
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						DIV0	UNALIGN	reserved				NOCP	INVPC	INVSTAT	UNDEF
Type	RO	RO	RO	RO	RO	RO	RW1C	RW1C	RO	RO	RO	RO	RW1C	RW1C	RW1C	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BFARV	reserved	BLSPERR	BSTKE	BUSTKE	IMPRE	PRECISE	IBUS	MMARV	reserved	MLSPERR	MSTKE	MUSTKE	reserved	DERR	IERR
Type	RW1C	RO	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RO	RW1C	RW1C	RW1C	RO	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
31:26	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
25	DIV0	RW1C	0	Divide-by-Zero Usage Fault Value Description 0 No divide-by-zero fault has occurred, or divide-by-zero trapping is not enabled. 1 The processor has executed an `SDIV` or `UDIV` instruction with a divisor of 0. When this bit is set, the **PC** value stacked for the exception return points to the instruction that performed the divide by zero. Trapping on divide-by-zero is enabled by setting the `DIV0` bit in the **Configuration and Control (CFGCTRL)** register (see page 157). This bit is cleared by writing a 1 to it.
24	UNALIGN	RW1C	0	Unaligned Access Usage Fault Value Description 0 No unaligned access fault has occurred, or unaligned access trapping is not enabled. 1 The processor has made an unaligned memory access. Unaligned `LDM`, `STM`, `LDRD`, and `STRD` instructions always fault regardless of the configuration of this bit. Trapping on unaligned access is enabled by setting the `UNALIGNED` bit in the **CFGCTRL** register (see page 157). This bit is cleared by writing a 1 to it.
23:20	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	NOCP	RW1C	0	No Coprocessor Usage Fault Value Description 0 A usage fault has not been caused by attempting to access a coprocessor. 1 The processor has attempted to access a coprocessor. This bit is cleared by writing a 1 to it.
18	INVPC	RW1C	0	Invalid PC Load Usage Fault Value Description 0 A usage fault has not been caused by attempting to load an invalid **PC** value. 1 The processor has attempted an illegal load of EXC_RETURN to the **PC** as a result of an invalid context or an invalid EXC_RETURN value. When this bit is set, the **PC** value stacked for the exception return points to the instruction that tried to perform the illegal load of the **PC**. This bit is cleared by writing a 1 to it.

Bit/Field	Name	Type	Reset	Description
17	INVSTAT	RW1C	0	Invalid State Usage Fault Value Description 0 A usage fault has not been caused by an invalid state. 1 The processor has attempted to execute an instruction that makes illegal use of the **EPSR** register. When this bit is set, the **PC** value stacked for the exception return points to the instruction that attempted the illegal use of the **Execution Program Status Register (EPSR)** register. This bit is not set if an undefined instruction uses the **EPSR** register. This bit is cleared by writing a 1 to it.
16	UNDEF	RW1C	0	Undefined Instruction Usage Fault Value Description 0 A usage fault has not been caused by an undefined instruction. 1 The processor has attempted to execute an undefined instruction. When this bit is set, the **PC** value stacked for the exception return points to the undefined instruction. An undefined instruction is an instruction that the processor cannot decode. This bit is cleared by writing a 1 to it.
15	BFARV	RW1C	0	Bus Fault Address Register Valid Value Description 0 The value in the **Bus Fault Address (FAULTADDR)** register is not a valid fault address. 1 The **FAULTADDR** register is holding a valid fault address. This bit is set after a bus fault, where the address is known. Other faults can clear this bit, such as a memory management fault occurring later. If a bus fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active bus fault handler whose **FAULTADDR** register value has been overwritten. This bit is cleared by writing a 1 to it.
14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	BLSPERR	RW1C	0	Bus Fault on Floating-Point Lazy State Preservation Value Description 0 No bus fault has occurred during floating-point lazy state preservation. 1 A bus fault has occurred during floating-point lazy state preservation. This bit is cleared by writing a 1 to it.

Bit/Field	Name	Type	Reset	Description
12	BSTKE	RW1C	0	Stack Bus Fault Value Description 0 No bus fault has occurred on stacking for exception entry. 1 Stacking for an exception entry has caused one or more bus faults. When this bit is set, the **SP** is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the **FAULTADDR** register. This bit is cleared by writing a 1 to it.
11	BUSTKE	RW1C	0	Unstack Bus Fault Value Description 0 No bus fault has occurred on unstacking for a return from exception. 1 Unstacking for a return from exception has caused one or more bus faults. This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The **SP** is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the **FAULTADDR** register. This bit is cleared by writing a 1 to it.
10	IMPRE	RW1C	0	Imprecise Data Bus Error Value Description 0 An imprecise data bus error has not occurred. 1 A data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error. When this bit is set, a fault address is not written to the **FAULTADDR** register. This fault is asynchronous. Therefore, if the fault is detected when the priority of the current process is higher than the bus fault priority, the bus fault becomes pending and becomes active only when the processor returns from all higher-priority processes. If a precise fault occurs before the processor enters the handler for the imprecise bus fault, the handler detects that both the `IMPRE` bit is set and one of the precise fault status bits is set. This bit is cleared by writing a 1 to it.
9	PRECISE	RW1C	0	Precise Data Bus Error Value Description 0 A precise data bus error has not occurred. 1 A data bus error has occurred, and the **PC** value stacked for the exception return points to the instruction that caused the fault. When this bit is set, the fault address is written to the **FAULTADDR** register. This bit is cleared by writing a 1 to it.

Bit/Field	Name	Type	Reset	Description
8	IBUS	RW1C	0	Instruction Bus Error Value Description 0 An instruction bus error has not occurred. 1 An instruction bus error has occurred. The processor detects the instruction bus error on prefetching an instruction, but sets this bit only if it attempts to issue the faulting instruction. When this bit is set, a fault address is not written to the **FAULTADDR** register. This bit is cleared by writing a 1 to it.
7	MMARV	RW1C	0	Memory Management Fault Address Register Valid Value Description 0 The value in the **Memory Management Fault Address (MMADDR)** register is not a valid fault address. 1 The **MMADDR** register is holding a valid fault address. If a memory management fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active memory management fault handler whose **MMADDR** register value has been overwritten. This bit is cleared by writing a 1 to it.
6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	MLSPERR	RW1C	0	Memory Management Fault on Floating-Point Lazy State Preservation Value Description 0 No memory management fault has occurred during floating-point lazy state preservation. 1 No memory management fault has occurred during floating-point lazy state preservation. This bit is cleared by writing a 1 to it.
4	MSTKE	RW1C	0	Stack Access Violation Value Description 0 No memory management fault has occurred on stacking for exception entry. 1 Stacking for an exception entry has caused one or more access violations. When this bit is set, the **SP** is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the **MMADDR** register. This bit is cleared by writing a 1 to it.

Bit/Field	Name	Type	Reset	Description
3	MUSTKE	RW1C	0	Unstack Access Violation Value Description 0 No memory management fault has occurred on unstacking for a return from exception. 1 Unstacking for a return from exception has caused one or more access violations. This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The **SP** is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the **MMADDR** register. This bit is cleared by writing a 1 to it.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	DERR	RW1C	0	Data Access Violation Value Description 0 A data access violation has not occurred. 1 The processor attempted a load or store at a location that does not permit the operation. When this bit is set, the **PC** value stacked for the exception return points to the faulting instruction and the address of the attempted access is written to the **MMADDR** register. This bit is cleared by writing a 1 to it.
0	IERR	RW1C	0	Instruction Access Violation Value Description 0 An instruction access violation has not occurred. 1 The processor attempted an instruction fetch from a location that does not permit execution. This fault occurs on any access to an XN region, even when the MPU is disabled or not present. When this bit is set, the **PC** value stacked for the exception return points to the faulting instruction and the address of the attempted access is not written to the **MMADDR** register. This bit is cleared by writing a 1 to it.

Register 77: Hard Fault Status (HFAULTSTAT), offset 0xD2C

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

The **HFAULTSTAT** register gives information about events that activate the hard fault handler.

Bits are cleared by writing a 1 to them.

Hard Fault Status (HFAULTSTAT)

Base 0xE000.E000
Offset 0xD2C
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DBG	FORCED	reserved													
Type	RW1C	RW1C	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														VECT	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW1C	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	DBG	RW1C	0	Debug Event This bit is reserved for Debug use. This bit must be written as a 0, otherwise behavior is unpredictable.
30	FORCED	RW1C	0	Forced Hard Fault Value Description 0 No forced hard fault has occurred. 1 A forced hard fault has been generated by escalation of a fault with configurable priority that cannot be handled, either because of priority or because it is disabled. When this bit is set, the hard fault handler must read the other fault status registers to find the cause of the fault. This bit is cleared by writing a 1 to it.
29:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	VECT	RW1C	0	Vector Table Read Fault Value Description 0 No bus fault has occurred on a vector table read. 1 A bus fault occurred on a vector table read. This error is always handled by the hard fault handler. When this bit is set, the **PC** value stacked for the exception return points to the instruction that was preempted by the exception. This bit is cleared by writing a 1 to it.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 78: Memory Management Fault Address (MMADDR), offset 0xD34

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

The **MMADDR** register contains the address of the location that generated a memory management fault. When an unaligned access faults, the address in the **MMADDR** register is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size. Bits in the **Memory Management Fault Status (MFAULTSTAT)** register indicate the cause of the fault and whether the value in the **MMADDR** register is valid (see page 166).

Memory Management Fault Address (MMADDR)

Base 0xE000.E000
Offset 0xD34
Type RW, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:0	ADDR	RW	-	Fault Address When the MMARV bit of **MFAULTSTAT** is set, this field holds the address of the location that generated the memory management fault.

Register 79: Bus Fault Address (FAULTADDR), offset 0xD38

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

The **FAULTADDR** register contains the address of the location that generated a bus fault. When an unaligned access faults, the address in the **FAULTADDR** register is the one requested by the instruction, even if it is not the address of the fault. Bits in the **Bus Fault Status (BFAULTSTAT)** register indicate the cause of the fault and whether the value in the **FAULTADDR** register is valid (see page 166).

Bus Fault Address (FAULTADDR)

Base 0xE000.E000
Offset 0xD38
Type RW, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:0	ADDR	RW	-	Fault Address When the FAULTADDRV bit of **BFAULTSTAT** is set, this field holds the address of the location that generated the bus fault.

3.6 Memory Protection Unit (MPU) Register Descriptions

This section lists and describes the Memory Protection Unit (MPU) registers, in numerical order by address offset.

The MPU registers can only be accessed from privileged mode.

Register 80: MPU Type (MPUTYPE), offset 0xD90

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

The **MPUTYPE** register indicates whether the MPU is present, and if so, how many regions it supports.

MPU Type (MPUTYPE)

Base 0xE000.E000
Offset 0xD90
Type RO, reset 0x0000.0800

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								IREGION							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DREGION								reserved							SEPARATE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	1	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.
23:16	IREGION	RO	0x00	Number of I Regions This field indicates the number of supported MPU instruction regions. This field always contains 0x00. The MPU memory map is unified and is described by the `DREGION` field.
15:8	DREGION	RO	0x08	Number of D Regions Value Description 0x08 Indicates there are eight supported MPU data regions.
7:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	SEPARATE	RO	0	Separate or Unified MPU Value Description 0 Indicates the MPU is unified.

Register 81: MPU Control (MPUCTRL), offset 0xD94

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

The **MPUCTRL** register enables the MPU, enables the default memory map background region, and enables use of the MPU when in the hard fault, Non-maskable Interrupt (NMI), and **Fault Mask Register (FAULTMASK)** escalated handlers.

When the ENABLE and PRIVDEFEN bits are both set:

- For privileged accesses, the default memory map is as described in “Memory Model” on page 81. Any access by privileged software that does not address an enabled memory region behaves as defined by the default memory map.
- Any access by unprivileged software that does not address an enabled memory region causes a memory management fault.

Execute Never (XN) and Strongly Ordered rules always apply to the System Control Space regardless of the value of the ENABLE bit.

When the ENABLE bit is set, at least one region of the memory map must be enabled for the system to function unless the PRIVDEFEN bit is set. If the PRIVDEFEN bit is set and no regions are enabled, then only privileged software can operate.

When the ENABLE bit is clear, the system uses the default memory map, which has the same memory attributes as if the MPU is not implemented (see Table 2-5 on page 84 for more information). The default memory map applies to accesses from both privileged and unprivileged software.

When the MPU is enabled, accesses to the System Control Space and vector table are always permitted. Other areas are accessible based on regions and whether PRIVDEFEN is set.

Unless HFNMIENA is set, the MPU is not enabled when the processor is executing the handler for an exception with priority –1 or –2. These priorities are only possible when handling a hard fault or NMI exception or when **FAULTMASK** is enabled. Setting the HFNMIENA bit enables the MPU when operating with these two priorities.

MPU Control (MPUCTRL)

Base 0xE000.E000
Offset 0xD94
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													PRIVDEFEN	HFNMIENA	ENABLE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2	PRIVDEFEN	RW	0	MPU Default Region This bit enables privileged software access to the default memory map. Value Description 0 If the MPU is enabled, this bit disables use of the default memory map. Any memory access to a location not covered by any enabled region causes a fault. 1 If the MPU is enabled, this bit enables use of the default memory map as a background region for privileged software accesses. When this bit is set, the background region acts as if it is region number -1. Any region that is defined and enabled has priority over this default map. If the MPU is disabled, the processor ignores this bit.
1	HFNMIENA	RW	0	MPU Enabled During Faults This bit controls the operation of the MPU during hard fault, NMI, and **FAULTMASK** handlers. Value Description 0 The MPU is disabled during hard fault, NMI, and **FAULTMASK** handlers, regardless of the value of the `ENABLE` bit. 1 The MPU is enabled during hard fault, NMI, and **FAULTMASK** handlers. When the MPU is disabled and this bit is set, the resulting behavior is unpredictable.
0	ENABLE	RW	0	MPU Enable Value Description 0 The MPU is disabled. 1 The MPU is enabled. When the MPU is disabled and the `HFNMIENA` bit is set, the resulting behavior is unpredictable.

Register 82: MPU Region Number (MPUNUMBER), offset 0xD98

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

The **MPUNUMBER** register selects which memory region is referenced by the **MPU Region Base Address (MPUBASE)** and **MPU Region Attribute and Size (MPUATTR)** registers. Normally, the required region number should be written to this register before accessing the **MPUBASE** or the **MPUATTR** register. However, the region number can be changed by writing to the **MPUBASE** register with the `VALID` bit set (see page 179). This write updates the value of the `REGION` field.

MPU Region Number (MPUNUMBER)

Base 0xE000.E000
Offset 0xD98
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													NUMBER		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	NUMBER	RW	0x0	MPU Region to Access This field indicates the MPU region referenced by the **MPUBASE** and **MPUATTR** registers. The MPU supports eight memory regions.

Register 83: MPU Region Base Address (MPUBASE), offset 0xD9C

Register 84: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4

Register 85: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC

Register 86: MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4

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

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

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

$N = \text{Log}_2\text{(Region size in bytes)}$

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

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

MPU Region Base Address (MPUBASE)

Base 0xE000.E000
Offset 0xD9C
Type RW, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADDR											VALID	reserved	REGION		
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	WO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:5	ADDR	RW	0x0000.000	Base Address Mask Bits 31:*N* in this field contain the region base address. The value of *N* depends on the region size, as shown above. The remaining bits (*N*-1):5 are reserved. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
4	VALID	WO	0	Region Number Valid Value Description 0 The **MPUNUMBER** register is not changed and the processor updates the base address for the region specified in the **MPUNUMBER** register and ignores the value of the `REGION` field. 1 The **MPUNUMBER** register is updated with the value of the `REGION` field and the base address is updated for the region specified in the `REGION` field. This bit is always read as 0.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	REGION	RW	0x0	Region Number On a write, contains the value to be written to the **MPUNUMBER** register. On a read, returns the current region number in the **MPUNUMBER** register.

Register 87: MPU Region Attribute and Size (MPUATTR), offset 0xDA0

Register 88: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8

Register 89: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0

Register 90: MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8

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

The **MPUATTR** register defines the region size and memory attributes of the MPU region specified by the **MPU Region Number (MPUNUMBER)** register and enables that region and any subregions.

The **MPUATTR** register is accessible using word or halfword accesses with the most-significant halfword holding the region attributes and the least-significant halfword holds the region size and the region and subregion enable bits.

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

The SIZE field defines the size of the MPU memory region specified by the **MPUNUMBER** register as follows:

(Region size in bytes) = $2^{(SIZE+1)}$

The smallest permitted region size is 32 bytes, corresponding to a SIZE value of 4. Table 3-10 on page 181 gives example SIZE values with the corresponding region size and value of N in the **MPU Region Base Address (MPUBASE)** register.

Table 3-10. Example SIZE Field Values

SIZE Encoding	Region Size	Value of N[a]	Note
00100b (0x4)	32 B	5	Minimum permitted size
01001b (0x9)	1 KB	10	-
10011b (0x13)	1 MB	20	-
11101b (0x1D)	1 GB	30	-
11111b (0x1F)	4 GB	No valid ADDR field in **MPUBASE**; the region occupies the complete memory map.	Maximum possible size

[a] Refers to the N parameter in the **MPUBASE** register (see page 179).

MPU Region Attribute and Size (MPUATTR)

Base 0xE000.E000
Offset 0xDA0
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			XN	reserved	AP			reserved		TEX			S	C	B
Type	RO	RO	RO	RW	RO	RW	RW	RW	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SRD								reserved		SIZE					ENABLE
Type	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	XN	RW	0	Instruction Access Disable Value Description 0 Instruction fetches are enabled. 1 Instruction fetches are disabled.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26:24	AP	RW	0	Access Privilege For information on using this bit field, see Table 3-5 on page 118.
23:22	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
21:19	TEX	RW	0x0	Type Extension Mask For information on using this bit field, see Table 3-3 on page 117.
18	S	RW	0	Shareable For information on using this bit, see Table 3-3 on page 117.
17	C	RW	0	Cacheable For information on using this bit, see Table 3-3 on page 117.
16	B	RW	0	Bufferable For information on using this bit, see Table 3-3 on page 117.
15:8	SRD	RW	0x00	Subregion Disable Bits Value Description 0 The corresponding subregion is enabled. 1 The corresponding subregion is disabled. Region sizes of 128 bytes and less do not support subregions. When writing the attributes for such a region, configure the SRD field as 0x00. See the section called “Subregions” on page 117 for more information.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:1	SIZE	RW	0x0	Region Size Mask The SIZE field defines the size of the MPU memory region specified by the **MPUNUMBER** register. Refer to Table 3-10 on page 181 for more information.

Bit/Field	Name	Type	Reset	Description
0	ENABLE	RW	0	Region Enable Value Description 0 The region is disabled. 1 The region is enabled.

3.7 Floating-Point Unit (FPU) Register Descriptions

This section lists and describes the Floating-Point Unit (FPU) registers, in numerical order by address offset.

Register 91: Coprocessor Access Control (CPAC), offset 0xD88

The **CPAC** register specifies the access privileges for coprocessors.

Coprocessor Access Control (CPAC)

Base 0xE000.E000
Offset 0xD88
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								CP11		CP10		reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	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
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:22	CP11	RW	0x00	CP11 Coprocessor Access Privilege Value Description 0x0 Access Denied Any attempted access generates a NOCP Usage Fault. 0x1 Privileged Access Only An unprivileged access generates a NOCP fault. 0x2 Reserved The result of any access is unpredictable. 0x3 Full Access
21:20	CP10	RW	0x00	CP10 Coprocessor Access Privilege Value Description 0x0 Access Denied Any attempted access generates a NOCP Usage Fault. 0x1 Privileged Access Only An unprivileged access generates a NOCP fault. 0x2 Reserved The result of any access is unpredictable. 0x3 Full Access
19:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 92: Floating-Point Context Control (FPCC), offset 0xF34

The **FPCC** register sets or returns FPU control data.

Floating-Point Context Control (FPCC)

Base 0xE000.E000
Offset 0xF34
Type RW, reset 0xC000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ASPEN	LSPEN	reserved													
Type	RW	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							MONRDY	reserved	BFRDY	MMRDY	HFRDY	THREAD	reserved	USER	LSPACT
Type	RO	RO	RO	RO	RO	RO	RO	RW	RO	RW	RW	RW	RW	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	ASPEN	RW	1	Automatic State Preservation Enable When set, enables the use of the FRACTV bit in the **CONTROL** register on execution of a floating-point instruction. This results in automatic hardware state preservation and restoration, for floating-point context, on exception entry and exit. **Important:** Two bits control when FPCA can be enabled: the ASPEN bit in the **Floating-Point Context Control (FPCC)** register and the DISFPCA bit in the **Auxiliary Control (ACTLR)** register.
30	LSPEN	RW	1	Lazy State Preservation Enable When set, enables automatic lazy state preservation for floating-point context.
29:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MONRDY	RW	0	Monitor Ready When set, DebugMonitor is enabled and priority permits setting MON_PEND when the floating-point stack frame was allocated.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BFRDY	RW	0	Bus Fault Ready When set, BusFault is enabled and priority permitted setting the BusFault handler to the pending state when the floating-point stack frame was allocated.
5	MMRDY	RW	0	Memory Management Fault Ready When set, MemManage is enabled and priority permitted setting the MemManage handler to the pending state when the floating-point stack frame was allocated.

Bit/Field	Name	Type	Reset	Description
4	HFRDY	RW	0	Hard Fault Ready When set, priority permitted setting the HardFault handler to the pending state when the floating-point stack frame was allocated.
3	THREAD	RW	0	Thread Mode When set, mode was Thread Mode when the floating-point stack frame was allocated.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	USER	RW	0	User Privilege Level When set, privilege level was user when the floating-point stack frame was allocated.
0	LSPACT	RW	0	Lazy State Preservation Active When set, Lazy State preservation is active. Floating-point stack frame has been allocated but saving state to it has been deferred.

Register 93: Floating-Point Context Address (FPCA), offset 0xF38

The **FPCA** register holds the location of the unpopulated floating-point register space allocated on an exception stack frame.

Floating-Point Context Address (FPCA)

Base 0xE000.E000
Offset 0xF38
Type RW, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:3	ADDRESS	RW	-	Address The location of the unpopulated floating-point register space allocated on an exception stack frame.
2:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 94: Floating-Point Default Status Control (FPDSC), offset 0xF3C

The **FPDSC** register holds the default values for the **Floating-Point Status Control (FPSC)** register.

Floating-Point Default Status Control (FPDSC)

Base 0xE000.E000
Offset 0xF3C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved					AHP	DN	FZ	RMODE		reserved					
Type	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	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
31:27	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	AHP	RW	-	AHP Bit Default This bit holds the default value for the `AHP` bit in the **FPSC** register.
25	DN	RW	-	DN Bit Default This bit holds the default value for the `DN` bit in the **FPSC** register.
24	FZ	RW	-	FZ Bit Default This bit holds the default value for the `FZ` bit in the **FPSC** register.
23:22	RMODE	RW	-	RMODE Bit Default This bit holds the default value for the `RMODE` bit field in the **FPSC** register. Value Description 0x0 Round to Nearest (RN) mode 0x1 Round towards Plus Infinity (RP) mode 0x2 Round towards Minus Infinity (RM) mode 0x3 Round towards Zero (RZ) mode
21:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

4 JTAG Interface

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

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

The TM4C1232C3PM JTAG controller works with the ARM JTAG controller built into the Cortex-M4F core by multiplexing the TDO outputs from both JTAG controllers. ARM JTAG instructions select the ARM TDO output while JTAG instructions select the TDO output. The multiplexer is controlled by the JTAG controller, which has comprehensive programming for the ARM, Tiva™ C Series microcontroller, and unimplemented JTAG instructions.

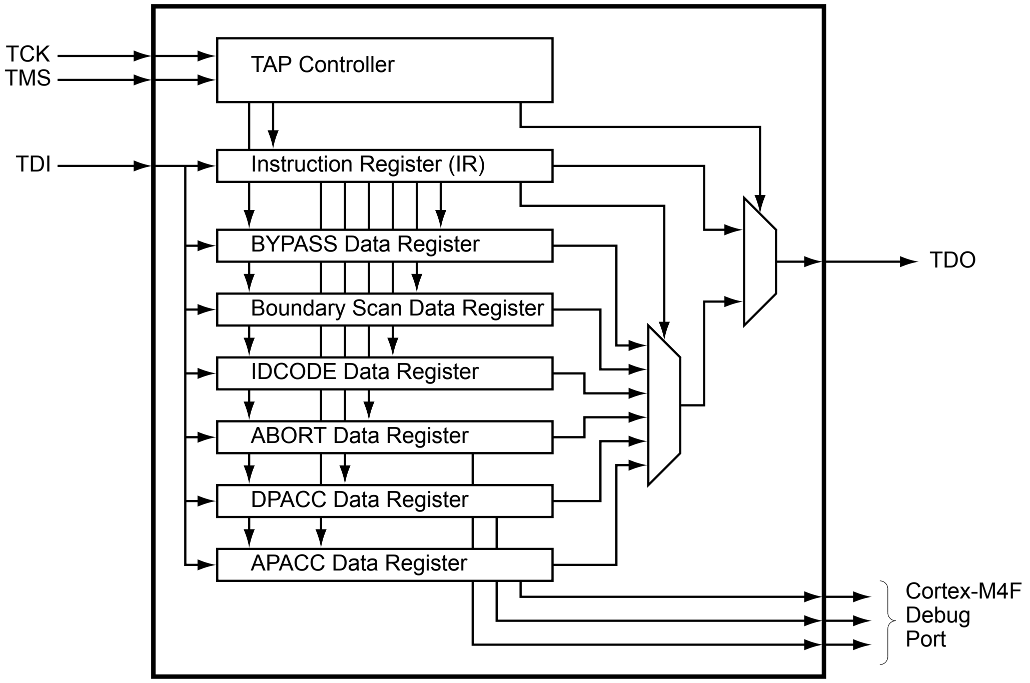
The TM4C1232C3PM JTAG module has the following features:

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

See the *ARM® Debug Interface V5 Architecture Specification* for more information on the ARM JTAG controller.

4.1 Block Diagram

Figure 4-1. JTAG Module Block Diagram



4.2 Signal Description

The following table lists the external signals of the JTAG/SWD controller and describes the function of each. The JTAG/SWD controller signals are alternate functions for some GPIO signals, however note that the reset state of the pins is for the JTAG/SWD function. The JTAG/SWD controller signals are under commit protection and require a special process to be configured as GPIOs, see “Commit Control” on page 588. 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 603) 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 620) 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 581.

Table 4-1. JTAG_SWD_SWO Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
SWCLK	52	PC0 (1)	I	TTL	JTAG/SWD CLK.
SWDIO	51	PC1 (1)	I/O	TTL	JTAG TMS and SWDIO.
SWO	49	PC3 (1)	O	TTL	JTAG TDO and SWO.
TCK	52	PC0 (1)	I	TTL	JTAG/SWD CLK.
TDI	50	PC2 (1)	I	TTL	JTAG TDI.
TDO	49	PC3 (1)	O	TTL	JTAG TDO and SWO.

Table 4-1. JTAG_SWD_SWO Signals (64LQFP) ***(continued)***

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
TMS	51	PC1 (1)	I	TTL	JTAG TMS and SWDIO.

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

4.3 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 4-1 on page 190. The JTAG module is composed of the Test Access Port (TAP) controller and serial shift chains with parallel update registers. The TAP controller is a simple state machine controlled by the TCK and TMS inputs. The current state of the TAP controller depends on the sequence of values captured on TMS at the rising edge of TCK. The TAP controller determines when the serial shift chains capture new data, shift data from TDI towards TDO, and update the parallel load registers. The current state of the TAP controller also determines whether the Instruction Register (IR) chain or one of the Data Register (DR) chains is being accessed.

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

Some instructions, like EXTEST, operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the BYPASS instruction to ensure that the serial path between TDI and TDO is always connected (see Table 4-3 on page 197 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 1127 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 202 for more information on reset.

4.3.1 JTAG Interface Pins

The JTAG interface consists of four standard pins: TCK, TMS, TDI, and TDO. These pins and their associated state after a power-on reset or reset caused by the $\overline{\text{RST}}$ input are given in Table 4-2. Detailed information on each pin follows.

Note: The following pins are configured as JTAG port pins out of reset. Refer to “General-Purpose Input/Outputs (GPIOs)” on page 581 for information on how to reprogram the configuration of these pins.

Table 4-2. JTAG Port Pins State after Power-On Reset or $\overline{\text{RST}}$ assertion

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

4.3.1.1 Test Clock Input (TCK)

The TCK pin is the clock for the JTAG module. This clock is provided so the test logic can operate independently of any other system clocks and to ensure that multiple JTAG TAP controllers that are daisy-chained together can synchronously communicate serial test data between components. During normal operation, TCK is driven by a free-running clock with a nominal 50% duty cycle. When necessary, TCK can be stopped at 0 or 1 for extended periods of time. While TCK is stopped at 0 or 1, the state of the TAP controller does not change and data in the JTAG Instruction and Data Registers is not lost.

By default, the internal pull-up resistor on the TCK pin is enabled after reset, assuring that no clocking occurs if the pin is not driven from an external source. The internal pull-up and pull-down resistors can be turned off to save internal power as long as the TCK pin is constantly being driven by an external source (see page 609 and page 611).

4.3.1.2 Test Mode Select (TMS)

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

Holding TMS high for five consecutive TCK cycles drives the TAP controller state machine to the Test-Logic-Reset state. When the TAP controller enters the Test-Logic-Reset state, the JTAG module and associated registers are reset to their default values. This procedure should be performed to initialize the JTAG controller. The JTAG Test Access Port state machine can be seen in its entirety in Figure 4-2 on page 193.

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 609).

4.3.1.3 Test Data Input (TDI)

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

By default, the internal pull-up resistor on the TDI pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on PC2/TDI; otherwise JTAG communication could be lost (see page 609).

4.3.1.4 Test Data Output (TDO)

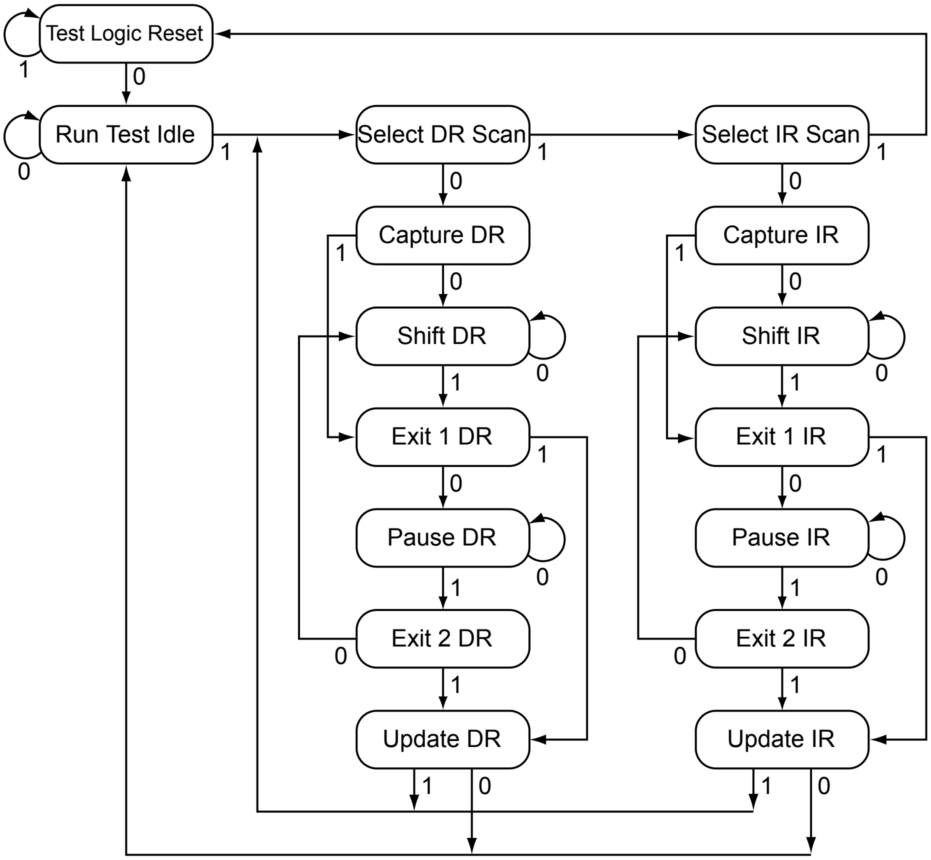
The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an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

By default, the internal pull-up resistor on the TDO pin is enabled after reset, assuring that the pin remains at a constant logic level when the JTAG port is not being used. The internal pull-up and pull-down resistors can be turned off to save internal power if a High-Z output value is acceptable during certain TAP controller states (see page 609 and page 611).

4.3.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 4-2. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR). In order to reset the JTAG module after the microcontroller has been powered on, the TMS input must be held HIGH for five TCK clock cycles, resetting the TAP controller and all associated JTAG chains. Asserting the correct sequence on the TMS pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, please refer to *IEEE Standard 1149.1*.

Figure 4-2. Test Access Port State Machine



4.3.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out on TDO during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on TDO, new data is being shifted into the serial shift register on TDI. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in “Register Descriptions” on page 197.

4.3.4 Operational Considerations

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

4.3.4.1 GPIO Functionality

When the microcontroller is reset with either a POR or $\overline{\text{RST}}$, the JTAG/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality (DEN[3:0] set in the **Port C GPIO Digital Enable (GPIODEN)** register), enabling the pull-up resistors (PUE[3:0] set in the **Port C GPIO Pull-Up Select (GPIOPUR)** register), disabling the pull-down resistors (PDE[3:0] cleared in the **Port C GPIO Pull-Down Select (GPIOPDR)** register) and enabling the alternate hardware function (AFSEL[3:0] set in the **Port C GPIO Alternate Function Select (GPIOAFSEL)** register) on the JTAG/SWD pins. See page 603, page 609, page 611, and page 614.

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

Caution – It is possible to create a software sequence that prevents the debugger from connecting to the TM4C1232C3PM microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger. In the case that the software routine is not implemented and the device is locked out of the part, this issue can be solved by using the TM4C1232C3PM Flash Programmer "Unlock" feature. Please refer to LMFLASHPROGRAMMER on the TI web for more information.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see “Signal Tables” on page 1099 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 603), **GPIO Pull Up Select (GPIOPUR)** register (see page 609), **GPIO Pull-Down Select (GPIOPDR)** register (see page 611), and **GPIO Digital Enable (GPIODEN)** register (see page 614) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 616) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 617) have been set.

4.3.4.2 Communication with JTAG/SWD

Because the debug clock and the system clock can be running at different frequencies, care must be taken to maintain reliable communication with the JTAG/SWD interface. In the Capture-DR state, the result of the previous transaction, if any, is returned, together with a 3-bit ACK response. Software should check the ACK response to see if the previous operation has completed before initiating a new transaction. Alternatively, if the system clock is at least 8 times faster than the debug clock (TCK or SWCLK), the previous operation has enough time to complete and the ACK bits do not have to be checked.

4.3.4.3 Recovering a "Locked" Microcontroller

Note: Performing the sequence below restores the non-volatile registers discussed in “Non-Volatile Register Programming” on page 465 to their factory default values. The mass erase of the Flash memory caused by the sequence below occurs prior to the non-volatile registers being restored.

In addition, the EEPROM is erased and its wear-leveling counters are returned to factory default values when performing the sequence below.

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

1. Assert and hold the $\overline{\text{RST}}$ signal.
2. Apply power to the device.
3. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence on the section called “JTAG-to-SWD Switching” on page 196.
4. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence on the section called “SWD-to-JTAG Switching” on page 196.
5. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
6. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
7. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
8. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
9. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
10. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
11. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
12. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
13. Release the $\overline{\text{RST}}$ signal.
14. Wait 400 ms.
15. Power-cycle the microcontroller.

4.3.4.4 ARM Serial Wire Debug (SWD)

In order to seamlessly integrate the ARM Serial Wire Debug (SWD) functionality, a serial-wire debugger must be able to connect to the Cortex-M4F core without having to perform, or have any knowledge of, JTAG cycles. This integration is accomplished with a SWD preamble that is issued before the SWD session begins.

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

Stepping through this sequence of the TAP state machine enables the SWD interface and disables the JTAG interface. For more information on this operation and the SWD interface, see the *ARM® Debug Interface V5 Architecture Specification*.

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

JTAG-to-SWD Switching

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

1. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that both JTAG and SWD are in their reset states.

2. Send the 16-bit JTAG-to-SWD switch command, 0xE79E, on TMS/SWDIO.

3. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that if SWJ-DP was already in SWD mode before sending the switch sequence, the SWD goes into the line reset state.

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

SWD-to-JTAG Switching

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

1. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that both JTAG and SWD are in their reset states.

2. Send the 16-bit SWD-to-JTAG switch command, 0xE73C, on TMS/SWDIO.

3. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that if SWJ-DP was already in JTAG mode before sending the switch sequence, the JTAG goes into the Test Logic Reset state.

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

4.4 Initialization and Configuration

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

In addition to enabling the alternate functions, any other changes to the GPIO pad configurations on the four JTAG pins (`PC[3:0]`) should be returned to their default settings.

4.5 Register Descriptions

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

4.5.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain connected between the JTAG `TDI` and `TDO` pins with a parallel load register. When the TAP Controller is placed in the correct states, bits can be shifted into the IR. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the IR bits is shown in Table 4-3. A detailed explanation of each instruction, along with its associated Data Register, follows.

Table 4-3. JTAG Instruction Register Commands

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

4.5.1.1 EXTEST Instruction

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

4.5.1.2 SAMPLE/PRELOAD Instruction

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

While these samples of the inputs, outputs, and output enables are being shifted out of the Boundary Scan Data Register, new data is being shifted into the Boundary Scan Data Register from TDI. Once the new data has been shifted into the Boundary Scan Data Register, the data is saved in the parallel load registers when the TAP controller enters the Update DR state. This update of the parallel load register preloads data into the Boundary Scan Data Register that is associated with each input, output, and output enable. This preloaded data can be used with the EXTEST instruction to drive data into or out of the controller. See “Boundary Scan Data Register” on page 199 for more information.

4.5.1.3 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between TDI and TDO. This instruction provides read and write access to the ABORT Register of the ARM Debug Access Port (DAP). Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. See the “ABORT Data Register” on page 200 for more information.

4.5.1.4 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between TDI and TDO. This instruction provides read and write access to the DPACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to the ARM debug and status registers. See “DPACC Data Register” on page 200 for more information.

4.5.1.5 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between TDI and TDO. This instruction provides read and write access to the APACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. See “APACC Data Register” on page 200 for more information.

4.5.1.6 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between TDI and TDO. This instruction provides information on the manufacturer, part number, and version of the ARM core. This information can be used by testing equipment and debuggers to automatically configure input and output data streams. IDCODE is the default instruction loaded into the JTAG Instruction Register when a Power-On-Reset (POR) is asserted, or the Test-Logic-Reset state is entered. See “IDCODE Data Register” on page 199 for more information.

4.5.1.7 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between TDI and TDO. This instruction is used to create a minimum length serial path between the TDI and TDO ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. See “BYPASS Data Register” on page 199 for more information.

4.5.2 Data Registers

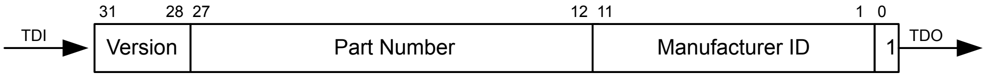
The JTAG module contains six Data Registers. These serial Data Register chains include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT and are discussed in the following sections.

4.5.2.1 IDCODE Data Register

The format for the 32-bit IDCODE Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 4-3. The standard requires that every JTAG-compliant microcontroller implement either the IDCODE instruction or the BYPASS instruction as the default instruction. The LSB of the IDCODE Data Register is defined to be a 1 to distinguish it from the BYPASS instruction, which has an LSB of 0. This definition allows auto-configuration test tools to determine which instruction is the default instruction.

The major uses of the JTAG port are for manufacturer testing of component assembly and program development and debug. To facilitate the use of auto-configuration debug tools, the IDCODE instruction outputs a value of 0x4BA0.0477. This value allows the debuggers to automatically configure themselves to work correctly with the Cortex-M4F during debug.

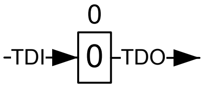
Figure 4-3. IDCODE Register Format



4.5.2.2 BYPASS Data Register

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

Figure 4-4. BYPASS Register Format

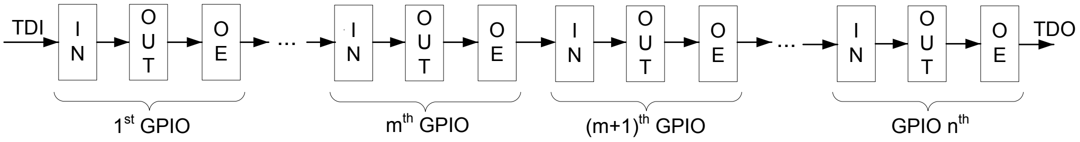


4.5.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 4-5. Each GPIO pin, starting with a GPIO pin next to the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as shown in the figure.

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

Figure 4-5. Boundary Scan Register Format



4.5.2.4 APACC Data Register

The format for the 35-bit APACC Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

4.5.2.5 DPACC Data Register

The format for the 35-bit DPACC Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

4.5.2.6 ABORT Data Register

The format for the 35-bit ABORT Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

5 System Control

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

5.1 Signal Description

The following table lists the external signals of the System Control module and describes the function of each. The NMI signal is the alternate function for two GPIO signals and functions as a GPIO after reset. The NMI pins are under commit protection and require a special process to be configured as any alternate function or to subsequently return to the GPIO function, see “Commit Control” on page 588. 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 603) 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 620) 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 581. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Table 5-1. System Control & Clocks Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
NMI	10 28	PD7 (8) PF0 (8)	I	TTL	Non-maskable interrupt.
OSC0	40	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	41	fixed	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
$\overline{\text{RST}}$	38	fixed	I	TTL	System reset input.

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

5.2 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 201
- Local control, such as reset (see “Reset Control” on page 202), power (see “Power Control” on page 207) and clock control (see “Clock Control” on page 208)
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 215

5.2.1 Device Identification

Several read-only registers provide software with information on the microcontroller, such as version, part number, memory sizes, and peripherals present on the device. The **Device Identification 0 (DID0)** (page 225) and **Device Identification 1 (DID1)** (page 227) registers provide details about the device's version, package, temperature range, and so on. The Peripheral Present registers starting at System Control offset 0x300, such as the **Watchdog Timer Peripheral Present (PPWD)** register, provide information on how many of each type of module are included on the device. Finally, information about the capabilities of the on-chip peripherals are provided at offset 0xFC0 in each peripheral's register space in the Peripheral Properties registers, such as the **GPTM Peripheral Properties (GPTMPP)** register. Previous devices used the **Device Capabilities (DC0-DC9)** registers for information about the peripherals and their capabilities. These registers are present on this device for backward software capability, but provide no information about peripherals that were not available on older devices.

5.2.2 Reset Control

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

5.2.2.1 Reset Sources

The TM4C1232C3PM microcontroller has six sources of reset:

1. Power-on reset (POR) (see page 203).
2. External reset input pin ($\overline{\text{RST}}$) assertion (see page 204).
3. A brown-out detection that can be caused by any of the following events: (see page 205).
   - V_{DD} under BOR0. The trigger value is the highest V_{DD} voltage level for BOR0.
   - V_{DD} under BOR1. The trigger value is the highest V_{DD} voltage level for BOR1.
4. Software-initiated reset (with the software reset registers) (see page 206).
5. A watchdog timer reset condition violation (see page 206).
6. MOSC failure (see page 207).

Table 5-2 provides a summary of results of the various reset operations.

Table 5-2. 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	Pin Config Only	Yes
Software System Request Reset using the SYSRESREQ bit in the **APINT** register.	Yes	Pin Config Only	Yes
Software System Request Reset using the VECTRESET bit in the **APINT** register.	Yes	Pin Config Only	No
Software Peripheral Reset	No	Pin Config Only	Yes[a]
Watchdog Reset	Yes	Pin Config Only	Yes
MOSC Failure Reset	Yes	Pin Config Only	Yes

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

After a reset, the **Reset Cause (RESC)** register is set with the reset cause. The bits in this register are sticky and maintain their state across multiple reset sequences, except when an internal POR is the cause, in which case, all the bits in the **RESC** register are cleared except for the POR indicator. A bit in the **RESC** register can be cleared by writing a 0.

At any reset that resets the core, the user has the opportunity to direct the core to execute the ROM Boot Loader or the application in Flash memory by using any GPIO signal as configured in the **Boot Configuration (BOOTCFG)** register.

At reset, the following sequence is performed:

1. The **BOOTCFG** register is read. If the EN bit is clear, the ROM Boot Loader is executed.

2. In the ROM Boot Loader, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.

3. f then EN bit is set or the status doesn't match the specified polarity, the data at address 0x0000.0004 is read, and if the data at this address is 0xFFFF.FFFF, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.

4. If there is data at address 0x0000.0004 that is not 0xFFFF.FFFF, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

Note: If the device fails the initialization phase, it toggles the TDO output pin as an indication the device is not executing. This feature is provided for debug purposes.

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

5.2.2.2 Power-On Reset (POR)

Note: The JTAG controller can only be reset by the power-on reset.

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_{VDD_POK}). The microcontroller must be operating within the specified operating parameters when the on-chip power-on reset pulse is complete (see “Power and Brown-Out” on page 1129). 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 204.

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.

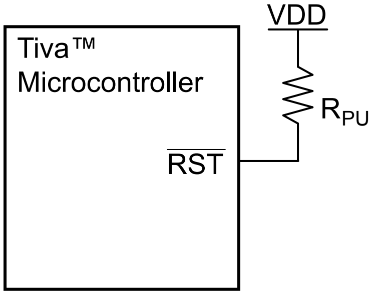
The internal POR is only active on the initial power-up of the microcontroller. The Power-On Reset timing is shown in “Power and Brown-Out” on page 1129.

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

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

If the application only uses the internal POR circuit, the $\overline{\text{RST}}$ input must be connected to the power supply (V_{DD}) through an optional pull-up resistor (0 to 100K Ω) as shown in Figure 5-1 on page 204. The $\overline{\text{RST}}$ input has filtering which requires a minimum pulse width in order for the reset pulse to be recognized, see Table 21-11 on page 1134.

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



R_{PU} = 0 to 100 kΩ

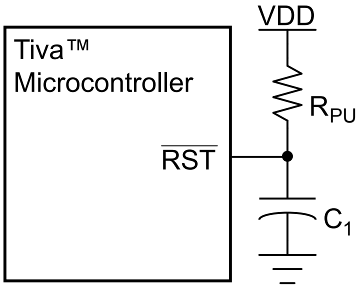
The external reset pin ($\overline{\text{RST}}$) resets the microcontroller including the core and all the on-chip peripherals. The external reset sequence is as follows:

1. The external reset pin ($\overline{\text{RST}}$) is asserted for the duration specified by T_{MIN} and then deasserted (see “Reset” on page 1134).

2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

To improve noise immunity and/or to delay reset at power up, the $\overline{\text{RST}}$ input may be connected to an RC network as shown in Figure 5-2 on page 204.

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

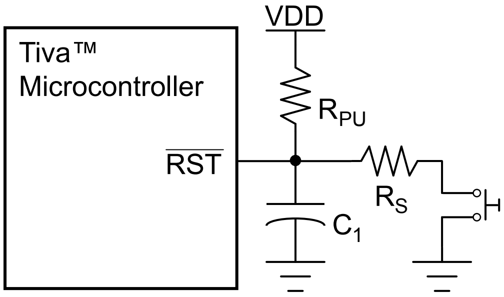


R_{PU} = 1 kΩ to 100 kΩ

C_1 = 1 nF to 10 µF

If the application requires the use of an external reset switch, Figure 5-3 on page 205 shows the proper circuitry to use.

Figure 5-3. Reset Circuit Controlled by Switch



Typical R_{PU} = 10 kΩ

Typical R_S = 470 Ω

C_1 = 10 nF

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

The external reset timing is shown in Figure 21-11 on page 1135.

5.2.2.4 Brown-Out Reset (BOR)

The microcontroller provides a brown-out detection circuit that triggers if any of the following occur:

- V_{DD} under BOR0. The external V_{DD} supply voltage is below the specified V_{DD} BOR0 value. The trigger value is the highest V_{DD} voltage level for BOR0.
- V_{DD} under BOR1. The external V_{DD} supply voltage is below the specified V_{DD} BOR1 value. The trigger value is the highest V_{DD} voltage level for BOR1.

The application can identify that a BOR event caused a reset by reading the **Reset Cause (RESC)** register. When a brown-out condition is detected, the default condition is to generate a reset. The BOR events can also be programmed to generate an interrupt by clearing the BOR0 bit or BOR1 bit in the **Power-On and Brown-Out Reset Control (PBORCTL)** register.

The brown-out reset sequence is as follows:

1. When V_{DD} drops below V_{BORnTH}, an internal BOR condition is set. Please refer to “Power and Brown-Out” on page 1129 for V_{BORnTH} value.
2. If the BOR condition exists, an internal reset is asserted.
3. The internal reset is released and the microcontroller fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.

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 “Power and Brown-Out” on page 1129.

5.2.2.5 Software Reset

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

Peripherals can be individually reset by software via peripheral-specific reset registers available beginning at System Control offset 0x500 (for example the **Watchdog Timer Software Reset (SRWD)** register). If the bit position corresponding to a peripheral is set and subsequently cleared, the peripheral is reset.

The entire microcontroller, including the core, can be reset by software by setting the SYSRESREQ bit in the **Application Interrupt and Reset Control (APINT)** register. The software-initiated system reset sequence is as follows:

1. A software microcontroller reset is initiated by setting the SYSRESREQ bit.
2. An internal reset is asserted.
3. The internal reset is deasserted and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The core only can be reset by software by setting the VECTRESET bit in the **APINT** register. The software-initiated core reset sequence is as follows:

1. A core reset is initiated by setting the VECTRESET bit.
2. An internal reset is asserted.
3. The internal reset is deasserted and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The software-initiated system reset timing is shown in Figure 21-12 on page 1135.

5.2.2.6 Watchdog Timer Reset

The Watchdog Timer module's function is to prevent system hangs. The TM4C1232C3PM 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 or a non-maskable 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 706.

The watchdog reset timing is shown in Figure 21-13 on page 1135.

5.2.3 Non-Maskable Interrupt

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

- The assertion of the NMI signal

- A main oscillator verification error

- The NMISET bit in the **Interrupt Control and State (INTCTRL)** register in the Cortex™-M4F (see page 149).

- The Watchdog module time-out interrupt when the INTTYPE bit in the **Watchdog Control (WDTCTL)** register is set (see page 712).

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

5.2.3.1 NMI Pin

The NMI signal is an alternate function for either GPIO port pin PD7 or PF0. 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 581. 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 617. The active sense of the NMI signal is High; asserting the enabled NMI signal above V_{IH} initiates the NMI interrupt sequence.

5.2.3.2 Main Oscillator Verification Failure

The TM4C1232C3PM microcontroller provides a main oscillator verification circuit that generates an error condition if the oscillator is running too fast or too slow. If the main oscillator verification circuit is enabled and a failure occurs, either a power-on reset is generated and control is transferred to the NMI handler, or an interrupt is generated. The MOSCIM bit in the **MOSCCTL** register determines which action occurs. In either case, the system clock source is automatically switched to the PIOSC. If a MOSC failure reset occurs, the NMI handler is used to address the main oscillator verification failure because the necessary code can be removed from the general reset handler, speeding up reset processing. The detection circuit is enabled by setting the CVAL bit in the **Main Oscillator Control (MOSCCTL)** register. The main oscillator verification error is indicated in the main oscillator fail status (MOSCFAIL) bit in the **Reset Cause (RESC)** register. The main oscillator verification circuit action is described in more detail in “Main Oscillator Verification Circuit” on page 214.

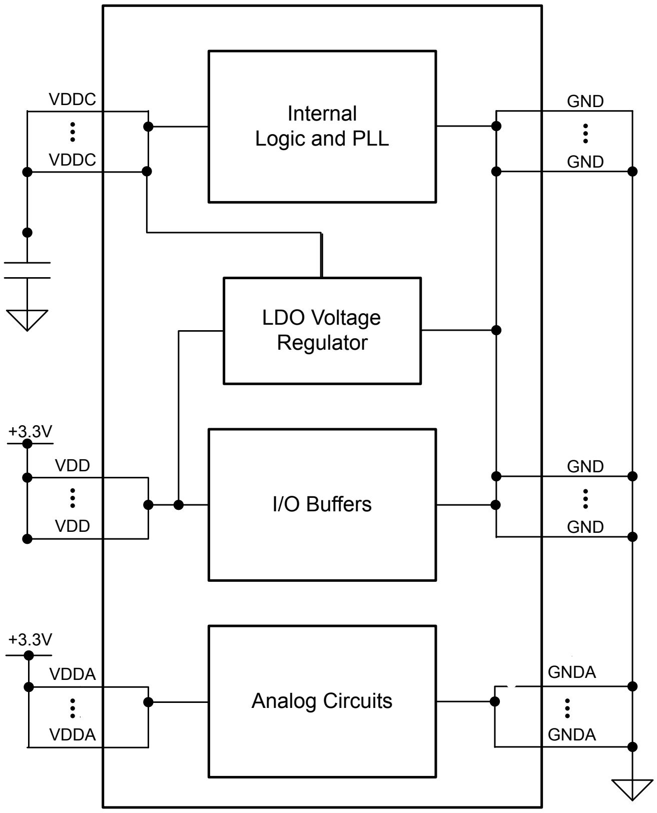
5.2.4 Power Control

The TM4C1232C3PM microcontroller provides an integrated LDO regulator that is used to provide power to the majority of the microcontroller's internal logic. Figure 5-4 shows the power architecture.

An external LDO may not be used.

Note: VDDA must be supplied with a voltage that meets the specification in Table 21-5 on page 1124, or the microcontroller does not function properly. VDDA is the supply for all of the analog circuitry on the device, including the clock circuitry.

Figure 5-4. Power Architecture



5.2.5 Clock Control

System control determines the control of clocks in this part.

5.2.5.1 Fundamental Clock Sources

There are multiple clock sources for use in the microcontroller:

- **Precision Internal Oscillator (PIOSC).** The precision internal oscillator is an on-chip clock source that is the clock source the microcontroller uses during and following POR. It does not require the use of any external components and provides a 16-MHz clock with ±1% accuracy with calibration and ±3% accuracy across temperature (see “PIOSC Specifications” on page 1139). 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. Regardless of whether or not the PIOSC is the source for the system clock, the PIOSC can be configured to be the source for the ADC clock as well as the baud clock for the UART and SSI, see “System Control” on page 215.

- **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 5 MHz to 25 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 4 MHz to 25 MHz. The single-ended clock source range is as specified in Table 21-13 on page 1138. The supported crystals are listed in the XTAL bit field in the **RCC** register (see page 241). Note that the MOSC provides the clock source for the USB PLL and must be connected to a crystal or an oscillator.

- **Low-Frequency Internal Oscillator (LFIOSC).** The low-frequency internal oscillator is intended for use during Deep-Sleep power-saving modes. The frequency can have wide variations; refer to "Low-Frequency Internal Oscillator (LFIOSC) Specifications" on page 1139 for more details. This power-savings mode benefits from reduced internal switching and also allows the MOSC to be powered down. In addition, the PIOSC can be powered down while in Deep-Sleep mode.

The internal system clock (SysClk), is derived from any of the above sources plus two others: the output of the main internal PLL and the precision internal oscillator divided by four (4 MHz ± 1%). The frequency of the PLL clock reference must be in the range of 5 MHz to 25 MHz (inclusive). Table 5-3 on page 209 shows how the various clock sources can be used in a system.

Table 5-3. Clock Source Options

Clock Source	Drive PLL?		Used as SysClk?	
Precision Internal Oscillator	Yes	BYPASS = 0, OSCSRC = 0x1	Yes	BYPASS = 1, OSCSRC = 0x1
Precision Internal Oscillator divide by 4 (4 MHz ± 1%)	No	-	Yes	BYPASS = 1, OSCSRC = 0x2
Main Oscillator	Yes	BYPASS = 0, OSCSRC = 0x0	Yes	BYPASS = 1, OSCSRC = 0x0
Low-Frequency Internal Oscillator (LFIOSC)	No	-	Yes	BYPASS = 1, OSCSRC = 0x3

5.2.5.2 Clock Configuration

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

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

Important: Write the **RCC** register prior to writing the **RCC2** register.

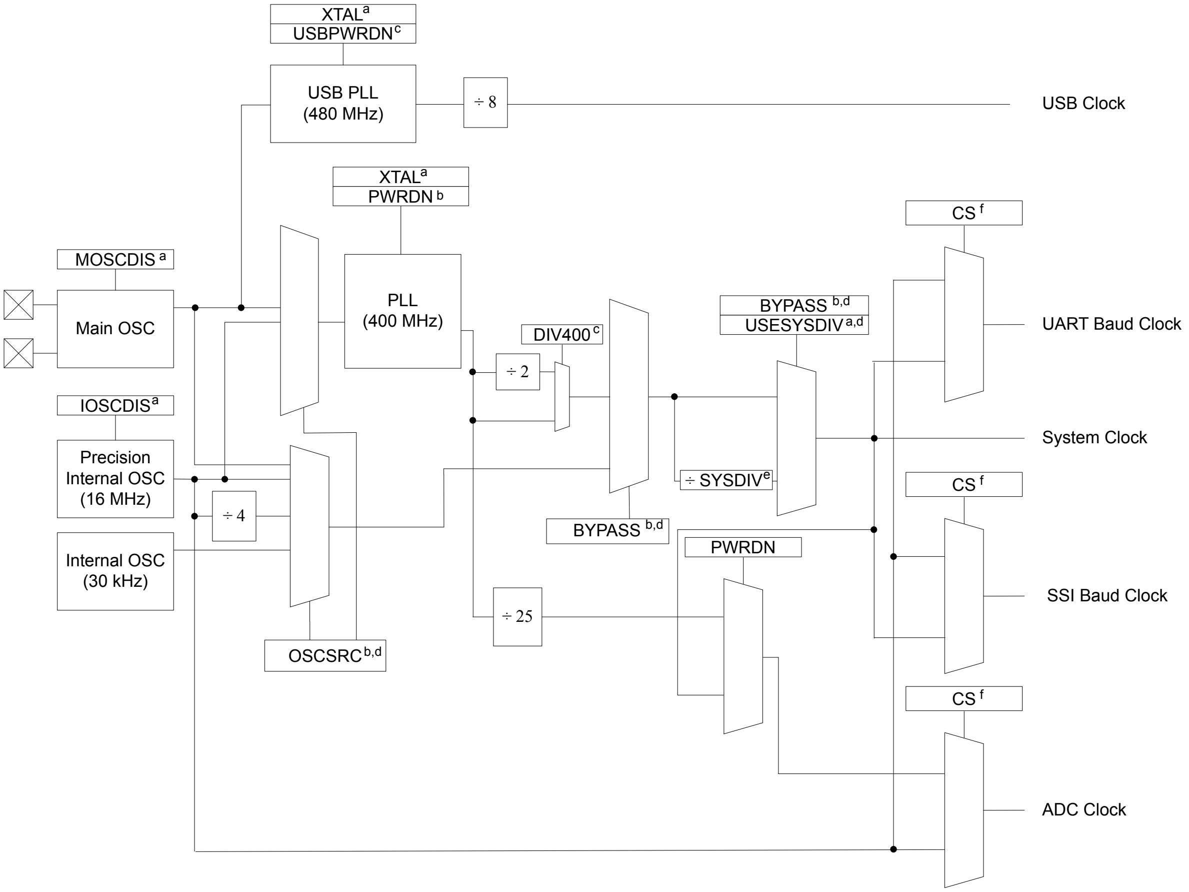
When transitioning the system clock configuration to use the MOSC as the fundamental clock source, the MOSCDIS bit must be set prior to reselecting the MOSC or an undefined system clock configuration can sporadically occur.

The configuration of the system clock must not be changed while an EEPROM operation is in process. Software must wait until the `WORKING` bit in the **EEPROM Done Status (EEDONE)** register is clear before making any changes to the system clock.

Figure 5-5 shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be individually enabled/disabled. The ADC clock signal can be selected from the PIOSC, the system clock if the PLL is disabled, or the PLL output divided down to 16 MHz if the PLL is enabled.

Note: If the ADC module is not using the PIOSC as the clock source, the system clock must be at least 16 MHz. When the USB module is in operation, MOSC must be the clock source, either with or without using the PLL, and the system clock must be at least 20 MHz.

Figure 5-5. Main Clock Tree



Note:

a. Control provided by **RCC** register bit/field.

b. Control provided by **RCC** register bit/field or **RCC2** register bit/field, if overridden with **RCC2** register bit `USERCC2`.

c. Control provided by **RCC2** register bit/field.

d. Also may be controlled by **DSLPCLKCFG** when in deep sleep mode.

e. Control provided by **RCC** register `SYSDIV` field, **RCC2** register `SYSDIV2` field if overridden with `USERCC2` bit, or [`SYSDIV2`,`SYSDIV2LSB`] if both `USERCC2` and `DIV400` bits are set.

f. Control provided by **UARTCC**, **SSICC**, and **ADCCC** register field.

Communication Clock Sources

In addition to the main clock tree described above, the UART, and SSI modules all have a Clock Control register in the peripheral's register map at offset 0xFC8 that can be used to select the clock source for the module's baud clock. Users can choose between the system clock, which is the default source for the baud clock, and the PIOSC. Note that there may be special considerations when using the PIOSC as the baud clock. For more information, see the Clock Control register description in the chapter describing the operation of the module.

Using the SYSDIV and SYSDIV2 Fields

In the **RCC** register, the SYSDIV field specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the BYPASS bit in this register is configured). When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. Table 5-4 shows how the SYSDIV encoding affects the system clock frequency, depending on whether the PLL is used (BYPASS=0) or another clock source is used (BYPASS=1). The divisor is equivalent to the SYSDIV encoding plus 1. For a list of possible clock sources, see Table 5-3 on page 209.

Table 5-4. Possible System Clock Frequencies Using the SYSDIV Field

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

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

The SYSDIV2 field in the **RCC2** register is 2 bits wider than the SYSDIV field in the **RCC** register so that additional larger divisors up to /64 are possible, allowing a lower system clock frequency for improved Deep Sleep power consumption. When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. The divisor is equivalent to the SYSDIV2 encoding plus 1. Table 5-5 shows how the SYSDIV2 encoding affects the system clock frequency, depending on whether the PLL is used (BYPASS2=0) or another clock source is used (BYPASS2=1). For a list of possible clock sources, see Table 5-3 on page 209.

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

SYSDIV2	Divisor	Frequency (BYPASS2=0)	Frequency (BYPASS2=1)	TivaWare Parameter[a]
0x00	/1	reserved	Clock source frequency/1	SYSCTL_SYSDIV_1
0x01	/2	reserved	Clock source frequency/2	SYSCTL_SYSDIV_2
0x02	/3	66.67 MHz	Clock source frequency/3	SYSCTL_SYSDIV_3
0x03	/4	50 MHz	Clock source frequency/4	SYSCTL_SYSDIV_4
0x04	/5	40 MHz	Clock source frequency/5	SYSCTL_SYSDIV_5
...	...	...	...	...
0x09	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
...	...	...	...	...
0x3F	/64	3.125 MHz	Clock source frequency/64	SYSCTL_SYSDIV_64

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

To allow for additional frequency choices when using the PLL, the DIV400 bit is provided along with the SYSDIV2LSB bit. When the DIV400 bit is set, bit 22 becomes the LSB for SYSDIV2. In this situation, the divisor is equivalent to the (SYSDIV2 encoding with SYSDIV2LSB appended) plus one. Table 5-6 shows the frequency choices when DIV400 is set. When the DIV400 bit is clear, SYSDIV2LSB is ignored, and the system clock frequency is determined as shown in Table 5-5 on page 212.

Table 5-6. Examples of Possible System Clock Frequencies with DIV400=1

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

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

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

5.2.5.3 Precision Internal Oscillator Operation (PIOSC)

The microcontroller powers up with the PIOSC running. If another clock source is desired, the PIOSC must remain enabled as it is used for internal functions. The PIOSC can only be disabled during Deep-Sleep mode. It can be powered down by setting the PIOSCPD bit in the **DSLPCLKCFG** register.

The PIOSC generates a 16-MHz clock with ±1% accuracy with calibration and ±3% accuracy across temperature (see “PIOSC Specifications” on page 1139). At the factory, the PIOSC is set to 16 MHz, however, the frequency can be trimmed for other voltage or temperature conditions using software in one of two 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.

5.2.5.4 Crystal Configuration for the Main Oscillator (MOSC)

The main oscillator supports the use of a select number of crystals from 4 to 25 MHz.

The XTAL bit in the **RCC** register (see page 241) describes the available crystal choices and default programming values.

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

5.2.5.5 Main PLL Frequency Configuration

The main PLL is disabled by default during power-on reset and is enabled later by software if required. Software specifies the output divisor to set the system clock frequency and enables the main PLL to drive the output. The PLL operates at 400 MHz, but is divided by two prior to the application of the output divisor, unless the DIV400 bit in the **RCC2** register is set.

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

If the main oscillator provides the clock reference to the main PLL, the translation provided by hardware and used to program the PLL is available for software in the **PLL Frequency n (PLLFREQn)** registers (see page 256). The internal translation provides a translation within ± 1% of the targeted PLL VCO frequency. Table 21-14 on page 1138 shows the actual PLL frequency and error for a given crystal choice.

The Crystal Value field (XTAL) in the **Run-Mode Clock Configuration (RCC)** register (see page 241) describes the available crystal choices and default programming of the **PLLFREQn** registers. Any time the XTAL field changes, the new settings are translated and the internal PLL settings are updated.

5.2.5.6 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: 5, 6, 8, 10, 12, 16, 18, 20, 24, or 25 MHz. Only these crystals provide the necessary USB PLL VCO frequency to conform with the USB timing specifications.

5.2.5.7 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 241 and page 247).

5.2.5.8 PLL Operation

If a PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is T_{READY} (see Table 21-13 on page 1138). During the relock time, the affected PLL is not usable as a clock reference. Software can poll the LOCK bit in the **PLL Status (PLLSTAT)** register to determine when the PLL has locked.

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 clocked by the system clock is used to measure the T_{READY} requirement. The down counter is set to 0x200 if the PLL is powering up. If the M or N values in the **PLLFREQn** registers are changed, the counter is set to 0xC0. 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.

5.2.5.9 Main Oscillator Verification Circuit

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

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

1. The MOSCFAIL bit in the **Reset Cause (RESC)** register is set.
2. The system clock is switched from the main oscillator to the PIOSC.
3. An internal power-on reset is initiated.
4. Reset is deasserted and the processor is directed to the NMI handler during the reset sequence.

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

1. The system clock is switched from the main oscillator to the PIOSC.

2. The MOFRIS bit in the **RIS** register is set to indicate a MOSC failure.

5.2.6 System Control

For power-savings purposes, the peripheral-specific **RCGCx**, **SCGCx**, and **DCGCx** registers (for example, **RCGCWD**) control the clock gating logic for that peripheral or block in the system while the microcontroller is in Run, Sleep, and Deep-Sleep mode, respectively. These registers are located in the System Control register map starting at offsets 0x600, 0x700, and 0x800, respectively. There must be a delay of 3 system clocks after a peripheral module clock is enabled in the **RCGC** register before any module registers are accessed.

Important: To support legacy software, the **RCGCn**, **SCGCn**, and **DCGCn** registers are available at offsets 0x100 - 0x128. A write to any of these legacy registers also writes the corresponding bit in the peripheral-specific **RCGCx**, **SCGCx**, and **DCGCx** registers. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. It is recommended that new software use the new registers and not rely on legacy operation.

If software uses a peripheral-specific register to write a legacy peripheral (such as TIMER0), the write causes proper operation, but the value of that bit is not reflected in the legacy register. Any bits that are changed by writing to a legacy register can be read back correctly with a read of the legacy register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

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

- Run mode
- Sleep mode
- Deep-Sleep mode

The following sections describe the different modes in detail.

Caution – If the Cortex-M4F Debug Access Port (DAP) has been enabled, and the device wakes from a low power sleep or deep-sleep mode, the core may start executing code before all clocks to peripherals have been restored to their Run mode configuration. The DAP is usually enabled by software tools accessing the JTAG or SWD interface when debugging or flash programming. If this condition occurs, a Hard Fault is triggered when software accesses a peripheral with an invalid clock.

A software delay loop can be used at the beginning of the interrupt routine that is used to wake up a system from a WFI (Wait For Interrupt) instruction. This stalls the execution of any code that accesses a peripheral register that might cause a fault. This loop can be removed for production software as the DAP is most likely not enabled during normal execution.

Because the DAP is disabled by default (power on reset), the user can also power cycle the device. The DAP is not enabled unless it is enabled through the JTAG or SWD interface.

5.2.6.1 Run Mode

In Run mode, the microcontroller actively executes code. Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the peripheral-specific **RCGC** registers. The system clock can be any of the available clock sources including the PLL.

5.2.6.2 Sleep Mode

In Sleep mode, the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code. Sleep mode is entered by the Cortex-M4F core executing a `WFI` (Wait for Interrupt) instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See “Power Management” on page 103 for more details.

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

Additional sleep modes are available that lower the power consumption of the SRAM and Flash memory. However, the lower power consumption modes have slower sleep and wake-up times, see “Dynamic Power Management” on page 217 for more information.

Important: Before executing the `WFI` instruction, software must confirm that the EEPROM is not busy by checking to see that the `WORKING` bit in the **EEPROM Done Status (EEDONE)** register is clear.

5.2.6.3 Deep-Sleep Mode

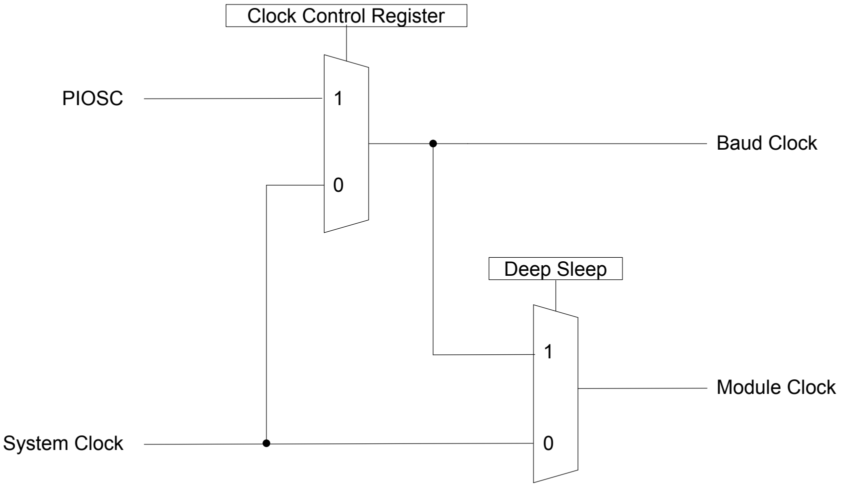
In Deep-Sleep mode, the clock frequency of the active peripherals may change (depending on the Deep-Sleep mode clock configuration) in addition to the processor clock being stopped. An interrupt returns the microcontroller to Run mode from one of the sleep modes; the sleep modes are entered on request from the code. Deep-Sleep mode is entered by first setting the `SLEEPDEEP` bit in the **System Control (SYSCTRL)** register (see page 155) and then executing a WFI instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See “Power Management” on page 103 for more details.

The Cortex-M4F processor core and the memory subsystem are not clocked in Deep-Sleep mode. Peripherals are clocked that are enabled in the peripheral-specific **DCGC** registers when auto-clock gating is enabled (see the **RCC** register) or the peripheral-specific **RCGC** registers when auto-clock gating is disabled. The system clock source is specified in the **DSLPCLKCFG** register. When the **DSLPCLKCFG** register is used, the internal oscillator source is powered up, if necessary, and other clocks are powered down. If the PLL is running at the time of the WFI instruction, hardware powers the PLL down and overrides the `SYSDIV` field of the active **RCC**/**RCC2** register, to be determined by the `DSDIVORIDE` setting in the **DSLPCLKCFG** register, up to /16 or /64 respectively. USB PLL is not powered down by execution of WFI instruction. When the Deep-Sleep exit event occurs, hardware brings the system clock back to the source and frequency it had at the onset of Deep-Sleep mode before enabling the clocks that had been stopped during the Deep-Sleep duration. If the PIOSC is used as the PLL reference clock source, it may continue to provide the clock during Deep-Sleep. See page 251.

Important: Before executing the `WFI` instruction, software must confirm that the EEPROM is not busy by checking to see that the `WORKING` bit in the **EEPROM Done Status (EEDONE)** register is clear.

To provide the lowest possible Deep-Sleep power consumption as well the ability to wake the processor from a peripheral without reconfiguring the peripheral for a change in clock, some of the communications modules have a Clock Control register at offset 0xFC8 in the module register space. The `CS` field in the Clock Control register allows the user to select the PIOSC as the clock source for the module's baud clock. When the microcontroller enters Deep-Sleep mode, the PIOSC becomes the source for the module clock as well, which allows the transmit and receive FIFOs to continue operation while the part is in Deep-Sleep. Figure 5-6 on page 217 shows how the clocks are selected.

Figure 5-6. Module Clock Selection



Additional deep-sleep modes are available that lower the power consumption of the SRAM and Flash memory. However, the lower power consumption modes have slower deep-sleep and wake-up times, see "Dynamic Power Management" on page 217 for more information.

5.2.6.4 Dynamic Power Management

In addition to the Sleep and Deep-Sleep modes and the clock gating for the on-chip modules, there are several additional power mode options that allow the LDO, Flash memory, and SRAM into different levels of power savings while in Sleep or Deep-Sleep modes. Note that these features may not be available on all devices; the **System Properties (SYSPROP)** register provides information on whether a mode is supported on a given MCU. The following registers provides these capabilities:

- **LDO Sleep Power Control (LDOSPCTL)**: controls the LDO value in Sleep mode
- **LDO Deep-Sleep Power Control (LDODPCTL)**: controls the LDO value in Deep-Sleep mode
- **LDO Sleep Power Calibration (LDOSPCAL)**: provides factory recommendations for the LDO value in Sleep mode
- **LDO Deep-Sleep Power Calibration (LDODPCAL)**: provides factory recommendations for the LDO value in Deep-Sleep mode
- **Sleep Power Configuration (SLPPWRCFG)**: controls the power saving modes for Flash memory and SRAM in Sleep mode
- **Deep-Sleep Power Configuration (DSLPPWRCFG)**: controls the power saving modes for Flash memory and SRAM in Deep-Sleep mode
- **Deep-Sleep Clock Configuration (DSLPCLKCFG)**: controls the clocking in Deep-Sleep mode
- **Sleep / Deep-Sleep Power Mode Status (SDPMST)**: provides status information on the various power saving events

LDO Sleep/Deep-Sleep Power Control

Note: While the device is connected through JTAG, the LDO control settings for Sleep or Deep-Sleep are not available and will not be applied.

The user can dynamically request to raise or lower the LDO voltage level to trade-off power/performance using either the **LDOSPCTL** register (see page 263) or the **LDODPCTL** register (see page 266). When lowering the LDO level, software must configure the system clock for the lower LDO value in **RCC**/**RCC2** for Sleep mode and in **DSLPCLKCFG** for Deep-Sleep mode before requesting the LDO to lower.

The LDO Power Calibration registers, **LDOSPCAL** and **LDODPCAL**, provide suggested values for the LDO in the various modes. If software requests an LDO value that is too low or too high, the value is not accepted and an error is reported in the **SDPMST** register.

The table below shows the maximum system clock frequency and PIOSC frequency with respect to the configured LDO voltage.

Operating Voltage (LDO)	Maximum System Clock Frequency	PIOSC
1.2	80 MHz	16 MHz
0.9	20 MHz	16 MHz

Flash Memory and SRAM Power Control

During Sleep or Deep-Sleep mode, Flash memory can be in either the default active mode or the low power mode; SRAM can be in the default active mode, standby mode, or low power mode. The active mode in each case provides the fastest times to sleep and wake up, but consumes more power. Low power mode provides the lowest power consumption, but takes longer to sleep and wake up.

The SRAM can be programmed to prohibit any power management by configuring the SRAMSM bit in the **System Properties (SYSPROP)** register. This configuration operates in the same way that legacy Stellaris® devices operate and provides the fastest sleep and wake-up times, but consumes the most power while in Sleep and Deep-Sleep mode. Other power options are retention mode, and retention mode with lower SRAM voltage. The SRAM retention mode with lower SRAM voltage provides the lowest power consumption, but has the longest sleep and wake-up times. These modes can be independently configured for Flash memory and SRAM using the **SLPPWRCFG** and **DSLPPWRCFG** registers.

The following power saving options are available in Sleep and Deep-Sleep modes:

- The clocks can be gated according to the settings in the the peripheral-specific **SCGC** or **DCGC** registers.
- In Deep-Sleep mode, the clock source can be changed and the PIOSC can be powered off (if no active peripheral requires it) using the **DSLPCLKCFG** register. These options are not available for Sleep mode.
- The LDO voltage can be changed using the **LDOSPCTL** or **LDODPCTL** register.
- The Flash memory can be put into low power mode. Refer to Table 21-23 on page 1144 for wake times from Sleep and Deep-Sleep.
- The SRAM can be put into standby or low power mode. Refer to Table 21-23 on page 1144 for wake times from Sleep and Deep-Sleep.

The **SDPMST** register provides results on the Dynamic Power Management command issued. It also has some real time status that can be viewed by a debugger or the core if it is running. These events do not trigger an interrupt and are meant to provide information to help tune software for power management. The status register gets written at the beginning of every Dynamic Power Management event request that provides error checking. There is no mechanism to clear the bits; they are overwritten on the next event. The real time data is real time and there is no event to register that information.

5.3 Initialization and Configuration

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

1. Bypass the PLL and system clock divider by setting the BYPASS bit and clearing the USESYS bit in the **RCC** register, thereby configuring the microcontroller to run off a "raw" clock source and allowing for the new PLL configuration to be validated before switching the system clock to the PLL.

2. Select the crystal value (XTAL) and oscillator source (OSCSRC), and clear the PWRDN bit in **RCC**/**RCC2**. Setting the XTAL field automatically pulls valid PLL configuration data for the appropriate crystal, and clearing the PWRDN bit powers and enables the PLL and its output.

3. Select the desired system divider (SYSDIV) in **RCC**/**RCC2** and set the USESYS bit in **RCC**. The SYSDIV field determines the system frequency for the microcontroller.

4. Wait for the PLL to lock by polling the PLLLRIS bit in the **Raw Interrupt Status (RIS**) register.

5. Enable use of the PLL by clearing the BYPASS bit in **RCC**/**RCC2**.

5.4 Register Map

Table 5-7 on page 219 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 457.

Table 5-7. System Control Register Map

Offset	Name	Type	Reset	Description	See page
System Control Registers					
0x000	DID0	RO	-	Device Identification 0	225
0x004	DID1	RO	0x1008.602E	Device Identification 1	227
0x030	PBORCTL	RW	0x0000.7FFF	Brown-Out Reset Control	230
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	231
0x054	IMC	RW	0x0000.0000	Interrupt Mask Control	234

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

Offset	Name	Type	Reset	Description	See page
0x058	MISC	RW1C	0x0000.0000	Masked Interrupt Status and Clear	236
0x05C	RESC	RW	-	Reset Cause	239
0x060	RCC	RW	0x0780.3AD1	Run-Mode Clock Configuration	241
0x06C	GPIOHBCTL	RW	0x0000.7E00	GPIO High-Performance Bus Control	245
0x070	RCC2	RW	0x07C0.6810	Run-Mode Clock Configuration 2	247
0x07C	MOSCCTL	RW	0x0000.0000	Main Oscillator Control	250
0x144	DSLPCLKCFG	RW	0x0780.0000	Deep Sleep Clock Configuration	251
0x14C	SYSPROP	RO	0x0000.1D31	System Properties	253
0x150	PIOSCCAL	RW	0x0000.0000	Precision Internal Oscillator Calibration	255
0x160	PLLFREQ0	RO	0x0000.0032	PLL Frequency 0	256
0x164	PLLFREQ1	RO	0x0000.0001	PLL Frequency 1	257
0x168	PLLSTAT	RO	0x0000.0000	PLL Status	258
0x188	SLPPWRCFG	RW	0x0000.0000	Sleep Power Configuration	259
0x18C	DSLPPWRCFG	RW	0x0000.0000	Deep-Sleep Power Configuration	261
0x1B4	LDOSPCTL	RW	0x0000.0018	LDO Sleep Power Control	263
0x1B8	LDOSPCAL	RO	0x0000.1818	LDO Sleep Power Calibration	265
0x1BC	LDODPCTL	RW	0x0000.0012	LDO Deep-Sleep Power Control	266
0x1C0	LDODPCAL	RO	0x0000.1212	LDO Deep-Sleep Power Calibration	268
0x1CC	SDPMST	RO	0x0000.0000	Sleep / Deep-Sleep Power Mode Status	269
0x300	PPWD	RO	0x0000.0003	Watchdog Timer Peripheral Present	272
0x304	PPTIMER	RO	0x0000.003F	16/32-Bit General-Purpose Timer Peripheral Present	273
0x308	PPGPIO	RO	0x0000.007F	General-Purpose Input/Output Peripheral Present	275
0x30C	PPDMA	RO	0x0000.0001	Micro Direct Memory Access Peripheral Present	278
0x314	PPHIB	RO	0x0000.0000	Hibernation Peripheral Present	279
0x318	PPUART	RO	0x0000.00FF	Universal Asynchronous Receiver/Transmitter Peripheral Present	280
0x31C	PPSSI	RO	0x0000.000F	Synchronous Serial Interface Peripheral Present	282
0x320	PPI2C	RO	0x0000.003F	Inter-Integrated Circuit Peripheral Present	284
0x328	PPUSB	RO	0x0000.0001	Universal Serial Bus Peripheral Present	286
0x334	PPCAN	RO	0x0000.0001	Controller Area Network Peripheral Present	287
0x338	PPADC	RO	0x0000.0003	Analog-to-Digital Converter Peripheral Present	288
0x33C	PPACMP	RO	0x0000.0001	Analog Comparator Peripheral Present	289

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

Offset	Name	Type	Reset	Description	See page
0x340	PPPWM	RO	0x0000.0000	Pulse Width Modulator Peripheral Present	290
0x344	PPQEI	RO	0x0000.0000	Quadrature Encoder Interface Peripheral Present	291
0x358	PPEEPROM	RO	0x0000.0001	EEPROM Peripheral Present	292
0x35C	PPWTIMER	RO	0x0000.003F	32/64-Bit Wide General-Purpose Timer Peripheral Present	293
0x500	SRWD	RW	0x0000.0000	Watchdog Timer Software Reset	295
0x504	SRTIMER	RW	0x0000.0000	16/32-Bit General-Purpose Timer Software Reset	297
0x508	SRGPIO	RW	0x0000.0000	General-Purpose Input/Output Software Reset	299
0x50C	SRDMA	RW	0x0000.0000	Micro Direct Memory Access Software Reset	301
0x518	SRUART	RW	0x0000.0000	Universal Asynchronous Receiver/Transmitter Software Reset	302
0x51C	SRSSI	RW	0x0000.0000	Synchronous Serial Interface Software Reset	304
0x520	SRI2C	RW	0x0000.0000	Inter-Integrated Circuit Software Reset	306
0x528	SRUSB	RW	0x0000.0000	Universal Serial Bus Software Reset	308
0x534	SRCAN	RW	0x0000.0000	Controller Area Network Software Reset	309
0x538	SRADC	RW	0x0000.0000	Analog-to-Digital Converter Software Reset	310
0x53C	SRACMP	RW	0x0000.0000	Analog Comparator Software Reset	312
0x558	SREEPROM	RW	0x0000.0000	EEPROM Software Reset	313
0x55C	SRWTIMER	RW	0x0000.0000	32/64-Bit Wide General-Purpose Timer Software Reset	314
0x600	RCGCWD	RW	0x0000.0000	Watchdog Timer Run Mode Clock Gating Control	316
0x604	RCGCTIMER	RW	0x0000.0000	16/32-Bit General-Purpose Timer Run Mode Clock Gating Control	317
0x608	RCGCGPIO	RW	0x0000.0000	General-Purpose Input/Output Run Mode Clock Gating Control	319
0x60C	RCGCDMA	RW	0x0000.0000	Micro Direct Memory Access Run Mode Clock Gating Control	321
0x618	RCGCUART	RW	0x0000.0000	Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control	322
0x61C	RCGCSSI	RW	0x0000.0000	Synchronous Serial Interface Run Mode Clock Gating Control	324
0x620	RCGCI2C	RW	0x0000.0000	Inter-Integrated Circuit Run Mode Clock Gating Control	326
0x628	RCGCUSB	RW	0x0000.0000	Universal Serial Bus Run Mode Clock Gating Control	328
0x634	RCGCCAN	RW	0x0000.0000	Controller Area Network Run Mode Clock Gating Control	329
0x638	RCGCADC	RW	0x0000.0000	Analog-to-Digital Converter Run Mode Clock Gating Control	330
0x63C	RCGCACMP	RW	0x0000.0000	Analog Comparator Run Mode Clock Gating Control	331

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

Offset	Name	Type	Reset	Description	See page
0x658	RCGCEEPROM	RW	0x0000.0000	EEPROM Run Mode Clock Gating Control	332
0x65C	RCGCWTIMER	RW	0x0000.0000	32/64-Bit Wide General-Purpose Timer Run Mode Clock Gating Control	333
0x700	SCGCWD	RW	0x0000.0000	Watchdog Timer Sleep Mode Clock Gating Control	335
0x704	SCGCTIMER	RW	0x0000.0000	16/32-Bit General-Purpose Timer Sleep Mode Clock Gating Control	336
0x708	SCGCGPIO	RW	0x0000.0000	General-Purpose Input/Output Sleep Mode Clock Gating Control	338
0x70C	SCGCDMA	RW	0x0000.0000	Micro Direct Memory Access Sleep Mode Clock Gating Control	340
0x718	SCGCUART	RW	0x0000.0000	Universal Asynchronous Receiver/Transmitter Sleep Mode Clock Gating Control	341
0x71C	SCGCSSI	RW	0x0000.0000	Synchronous Serial Interface Sleep Mode Clock Gating Control	343
0x720	SCGCI2C	RW	0x0000.0000	Inter-Integrated Circuit Sleep Mode Clock Gating Control	345
0x728	SCGCUSB	RW	0x0000.0000	Universal Serial Bus Sleep Mode Clock Gating Control	347
0x734	SCGCCAN	RW	0x0000.0000	Controller Area Network Sleep Mode Clock Gating Control	348
0x738	SCGCADC	RW	0x0000.0000	Analog-to-Digital Converter Sleep Mode Clock Gating Control	349
0x73C	SCGCACMP	RW	0x0000.0000	Analog Comparator Sleep Mode Clock Gating Control	350
0x758	SCGCEEPROM	RW	0x0000.0000	EEPROM Sleep Mode Clock Gating Control	351
0x75C	SCGCWTIMER	RW	0x0000.0000	32/64-Bit Wide General-Purpose Timer Sleep Mode Clock Gating Control	352
0x800	DCGCWD	RW	0x0000.0000	Watchdog Timer Deep-Sleep Mode Clock Gating Control	354
0x804	DCGCTIMER	RW	0x0000.0000	16/32-Bit General-Purpose Timer Deep-Sleep Mode Clock Gating Control	355
0x808	DCGCGPIO	RW	0x0000.0000	General-Purpose Input/Output Deep-Sleep Mode Clock Gating Control	357
0x80C	DCGCDMA	RW	0x0000.0000	Micro Direct Memory Access Deep-Sleep Mode Clock Gating Control	359
0x818	DCGCUART	RW	0x0000.0000	Universal Asynchronous Receiver/Transmitter Deep-Sleep Mode Clock Gating Control	360
0x81C	DCGCSSI	RW	0x0000.0000	Synchronous Serial Interface Deep-Sleep Mode Clock Gating Control	362
0x820	DCGCI2C	RW	0x0000.0000	Inter-Integrated Circuit Deep-Sleep Mode Clock Gating Control	364
0x828	DCGCUSB	RW	0x0000.0000	Universal Serial Bus Deep-Sleep Mode Clock Gating Control	366

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

Offset	Name	Type	Reset	Description	See page
0x834	DCGCCAN	RW	0x0000.0000	Controller Area Network Deep-Sleep Mode Clock Gating Control	367
0x838	DCGCADC	RW	0x0000.0000	Analog-to-Digital Converter Deep-Sleep Mode Clock Gating Control	368
0x83C	DCGCACMP	RW	0x0000.0000	Analog Comparator Deep-Sleep Mode Clock Gating Control	369
0x858	DCGCEEPROM	RW	0x0000.0000	EEPROM Deep-Sleep Mode Clock Gating Control	370
0x85C	DCGCWTIMER	RW	0x0000.0000	32/64-Bit Wide General-Purpose Timer Deep-Sleep Mode Clock Gating Control	371
0xA00	PRWD	RO	0x0000.0000	Watchdog Timer Peripheral Ready	373
0xA04	PRTIMER	RO	0x0000.0000	16/32-Bit General-Purpose Timer Peripheral Ready	374
0xA08	PRGPIO	RO	0x0000.0000	General-Purpose Input/Output Peripheral Ready	376
0xA0C	PRDMA	RO	0x0000.0000	Micro Direct Memory Access Peripheral Ready	378
0xA18	PRUART	RO	0x0000.0000	Universal Asynchronous Receiver/Transmitter Peripheral Ready	379
0xA1C	PRSSI	RO	0x0000.0000	Synchronous Serial Interface Peripheral Ready	381
0xA20	PRI2C	RO	0x0000.0000	Inter-Integrated Circuit Peripheral Ready	383
0xA28	PRUSB	RO	0x0000.0000	Universal Serial Bus Peripheral Ready	385
0xA34	PRCAN	RO	0x0000.0000	Controller Area Network Peripheral Ready	386
0xA38	PRADC	RO	0x0000.0000	Analog-to-Digital Converter Peripheral Ready	387
0xA3C	PRACMP	RO	0x0000.0000	Analog Comparator Peripheral Ready	388
0xA58	PREEPROM	RO	0x0000.0000	EEPROM Peripheral Ready	389
0xA5C	PRWTIMER	RO	0x0000.0000	32/64-Bit Wide General-Purpose Timer Peripheral Ready	390
System Control Legacy Registers					
0x008	DC0	RO	0x002F.000F	Device Capabilities 0	392
0x010	DC1	RO	0x1103.2FBF	Device Capabilities 1	394
0x014	DC2	RO	0x030F.F037	Device Capabilities 2	397
0x018	DC3	RO	0xBFFF.0FC0	Device Capabilities 3	400
0x01C	DC4	RO	0x0004.F07F	Device Capabilities 4	404
0x020	DC5	RO	0x0000.0000	Device Capabilities 5	407
0x024	DC6	RO	0x0000.0011	Device Capabilities 6	409
0x028	DC7	RO	0xFFFF.FFFF	Device Capabilities 7	410
0x02C	DC8	RO	0x0FFF.0FFF	Device Capabilities 8	413
0x040	SRCR0	RO	0x0000.0000	Software Reset Control 0	416

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

Offset	Name	Type	Reset	Description	See page
0x044	SRCR1	RO	0x0000.0000	Software Reset Control 1	418
0x048	SRCR2	RO	0x0000.0000	Software Reset Control 2	421
0x100	RCGC0	RO	0x0000.0040	Run Mode Clock Gating Control Register 0	423
0x104	RCGC1	RO	0x0000.0000	Run Mode Clock Gating Control Register 1	426
0x108	RCGC2	RO	0x0000.0000	Run Mode Clock Gating Control Register 2	429
0x110	SCGC0	RO	0x0000.0040	Sleep Mode Clock Gating Control Register 0	432
0x114	SCGC1	RO	0x0000.0000	Sleep Mode Clock Gating Control Register 1	434
0x118	SCGC2	RO	0x0000.0000	Sleep Mode Clock Gating Control Register 2	437
0x120	DCGC0	RO	0x0000.0040	Deep Sleep Mode Clock Gating Control Register 0	439
0x124	DCGC1	RO	0x0000.0000	Deep-Sleep Mode Clock Gating Control Register 1	441
0x128	DCGC2	RO	0x0000.0000	Deep Sleep Mode Clock Gating Control Register 2	444
0x190	DC9	RO	0x00FF.00FF	Device Capabilities 9	446
0x1A0	NVMSTAT	RO	0x0000.0001	Non-Volatile Memory Information	448

5.5 System Control Register Descriptions

All addresses given are relative to the System Control base address of 0x400F.E000. Registers provided for legacy software support only are listed in “System Control Legacy Register Descriptions” on page 391.

Register 1: Device Identification 0 (DID0), offset 0x000

This register identifies the version of the microcontroller. Each microcontroller is uniquely identified by the combined values of the `CLASS` field in the **DID0** register and the `PARTNO` field in the **DID1** register. The `MAJOR` and `MINOR` bit fields indicate the die revision number. Combined, the `MAJOR` and `MINOR` bit fields indicate the part revision number.

MAJOR Bitfield Value	MINOR Bitfield Value	Die Revision	Part Revision
0x0	0x0	A0	1
0x0	0x1	A1	2
0x0	0x2	A2	3
0x0	0x3	A3	4
0x1	0x0	B0	5
0x1	0x1	B1	6
0x1	0x2	B2	7

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	1	0	0	0	0	0	0	0	0	1	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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	0x01	DID0 Version This field defines the **DID0** register format version. The version number is numeric. The value of the `VER` field is encoded as follows (all other encodings are reserved): Value Description 0x1 Second version of the **DID0** register format.
27:24	reserved	RO	0x08	Software should not rely on the value of 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
23:16	CLASS	RO	0x05	Device Class The `CLASS` field value identifies the internal design from which all mask sets are generated for all microcontrollers in a particular product line. The `CLASS` field value is changed for new product lines, for changes in fab process (for example, a remap or shrink), or any case where the `MAJOR` or `MINOR` fields require differentiation from prior microcontrollers. The value of the `CLASS` field is encoded as follows (all other encodings are reserved): Value Description 0x05 Tiva™ TM4C123x microcontrollers
15:8	MAJOR	RO	-	Major Die Revision This field specifies the major revision number of the microcontroller. The major revision reflects changes to base layers of the design. This field is encoded as follows: Value Description 0x0 Revision A (initial device) 0x1 Revision B (first base layer revision) 0x2 Revision C (second base layer revision) and so on.
7:0	MINOR	RO	-	Minor Die Revision 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: Value Description 0x0 Initial device, or a major revision update. 0x1 First metal layer change. 0x2 Second metal layer change. and so on.

Register 2: Device Identification 1 (DID1), offset 0x004

This register identifies the device family, part number, temperature range, pin count, and package type. Each microcontroller is uniquely identified by the combined values of the CLASS field in the **DID0** register and the PARTNO field in the **DID1** register.

Device Identification 1 (DID1)

Base 0x400F.E000
Offset 0x004
Type RO, reset 0x1008.602E

	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	0	0	0	1	0	0	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	1	0	0	0	0	0	0	0	1	0	1	1	1	0

Bit/Field	Name	Type	Reset	Description
31:28	VER	RO	0x1	DID1 Version This field defines the **DID1** register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved): Value Description 0x0 Initial **DID1** register format definition, indicating a Stellaris LM3Snnn device. 0x1 Second version of the **DID1** register format.
27:24	FAM	RO	0x0	Family This field provides the family identification of the device within the product portfolio. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 Tiva™ C Series microcontrollers and legacy Stellaris microcontrollers, that is, all devices with external part numbers starting with TM4C, LM4F or LM3S.
23:16	PARTNO	RO	0x8	Part Number This field provides the part number of the device within the family. The reset value shown indicates the TM4C1232C3PM microcontroller.

Bit/Field	Name	Type	Reset	Description
15:13	PINCOUNT	RO	0x3	Package Pin Count This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 reserved 0x1 reserved 0x2 100-pin package 0x3 64-pin package 0x4 144-pin package 0x5 157-pin package 0x6 168-pin package
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	0x1	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 Reserved 0x1 Industrial temperature range (-40°C to 85°C) 0x2 Extended temperature range (-40°C to 105°C) 0x3 Available in both industrial temperature range (-40°C to 85°C) and extended temperature range (-40°C to 105°C) devices. See “Package Information” on page 1164 for specific order numbers.
4:3	PKG	RO	0x1	Package Type This field specifies the package type. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 Reserved 0x1 LQFP package 0x2 BGA package
2	ROHS	RO	0x1	RoHS-Compliance This bit specifies whether the device is RoHS-compliant. A 1 indicates the part is RoHS-compliant.

Bit/Field	Name	Type	Reset	Description
1:0	QUAL	RO	0x2	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 3: Brown-Out Reset Control (PBORCTL), offset 0x030

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

Note: The BOR voltage values and center points are based on simulation only. These values are yet to be characterized and are subject to change.

Brown-Out Reset Control (PBORCTL)

Base 0x400F.E000
Offset 0x030
Type RW, 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													BOR0	BOR1	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Bit/Field	Name	Type	Reset	Description
31:3	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	BOR0	RW	1	VDD under BOR0 Event Action The VDD BOR0 trip value is 3.02V +/- 90mv. Value Description 0 A BOR0 event causes an interrupt to be generated in the interrupt controller. 1 A BOR0 event causes a reset of the microcontroller.
1	BOR1	RW	1	VDD under BOR1 Event Action The VDD BOR1 trip value is 2.88V +/- 90mv. Value Description 0 A BOR1 event causes an interrupt to be generated to the interrupt controller. 1 A BOR1 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 4: Raw Interrupt Status (RIS), offset 0x050

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

Raw Interrupt Status (RIS)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				BOR0RIS	VDDARIS	reserved	MOSCPUPRIS	USBPLLLRIS	PLLLRIS	reserved		MOFRIS	reserved	BOR1RIS	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:12	reserved	RO	0x0000.00	Software should not rely on the value of 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	BOR0RIS	RO	0	VDD under BOR0 Raw Interrupt Status Value Description 0 A VDD BOR0 condition is not currently active. 1 A VDD BOR0 condition is currently active. Note the `BOR0` bit in the **PBORCTL** register must be cleared to cause an interrupt due to a BOR0 Event. This bit is cleared by writing a 1 to the `BOR0MIS` bit in the **MISC** register.
10	VDDARIS	RO	0	VDDA Power OK Event Raw Interrupt Status Value Description 0 VDDA power is not at its appropriate functional voltage. 1 VDDA is at an appropriate functional voltage. This bit is cleared by writing a 1 to the `VDDAMIS` bit in the **MISC** register.
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	MOSCPUPRIS	RO	0	MOSC Power Up Raw Interrupt Status Value Description 0 Sufficient time has not passed for the MOSC to reach the expected frequency. 1 Sufficient time has passed for the MOSC to reach the expected frequency. The value for this power-up time is indicated by T_{MOSC_START}. This bit is cleared by writing a 1 to the `MOSCPUPMIS` bit in the **MISC** register.
7	USBPLLLRIS	RO	0	USB PLL Lock Raw Interrupt Status Value Description 0 The USB PLL timer has not reached T_{READY}. 1 The USB PLL timer has reached T_{READY} indicating that sufficient time has passed for the USB PLL to lock. This bit is cleared by writing a 1 to the `USBPLLLMIS` bit in the **MISC** register.
6	PLLLRIS	RO	0	PLL Lock Raw Interrupt Status Value Description 0 The PLL timer has not reached T_{READY}. 1 The PLL timer has reached T_{READY} indicating that sufficient time has passed for the PLL to lock. This bit is cleared by writing a 1 to the `PLLLMIS` bit in the **MISC** register.
5:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MOFRIS	RO	0	Main Oscillator Failure Raw Interrupt Status Value Description 0 The main oscillator has not failed. 1 The `MOSCIM` bit in the **MOSCCTL** register is set and the main oscillator has failed. This bit is cleared by writing a 1 to the `MOFMIS` bit in the **MISC** register.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
1	BOR1RIS	RO	0	VDD under BOR1 Raw Interrupt Status Value Description 0 A VDDS BOR1 condition is not currently active. 1 A VDDS BOR1 condition is currently active. Note the `BOR1` bit in the **PBORCTL** register must be cleared to cause an interrupt due to a BOR1 Event. This bit is cleared by writing a 1 to the `BOR1MIS` bit in the **MISC** register.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

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

Interrupt Mask Control (IMC)

Base 0x400F.E000
Offset 0x054
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				BOR0IM	VDDAIM	reserved	MOSCPUPIM	USBPLLLIM	PLLLIM	reserved		MOFIM	reserved	BOR1IM	reserved
Type	RO	RO	RO	RO	RW	RW	RO	RW	RW	RW	RO	RO	RW	RO	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.00	Software should not rely on the value of 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	BOR0IM	RW	0	VDD under BOR0 Interrupt Mask Value Description 0 The `BOR0RIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `BOR0RIS` bit in the **RIS** register is set.
10	VDDAIM	RW	0	VDDA Power OK Interrupt Mask Value Description 0 The `VDDARIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `VDDARIS` bit in the **RIS** register is set.
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	MOSCPUPIM	RW	0	MOSC Power Up Interrupt Mask Value Description 0 The `MOSCPUPRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `MOSCPUPRIS` bit in the **RIS** register is set.

Bit/Field	Name	Type	Reset	Description
7	USBPLLLIM	RW	0	USB PLL Lock Interrupt Mask Value Description 0 The `USBPLLLRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `USBPLLLRIS` bit in the **RIS** register is set.
6	PLLLIM	RW	0	PLL Lock Interrupt Mask Value Description 0 The `PLLLRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `PLLLRIS` bit in the **RIS** register is set.
5:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MOFIM	RW	0	Main Oscillator Failure Interrupt Mask Value Description 0 The `MOFRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `MOFRIS` bit in the **RIS** register is set.
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BOR1IM	RW	0	VDD under BOR1 Interrupt Mask Value Description 0 The `BOR1RIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `BOR1RIS` bit in the **RIS** register is set.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

On a read, this register gives the current masked status value of the corresponding interrupt in the **Raw Interrupt Status (RIS)** register. All of the bits are RW1C, thus writing a 1 to a bit clears the corresponding raw interrupt bit in the **RIS** register (see page 231).

Masked Interrupt Status and Clear (MISC)

Base 0x400F.E000
Offset 0x058
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				BOR0MIS	VDDAMIS	reserved	MOSCPUPMIS	USBPLLLMIS	PLLLMIS	reserved		MOFMIS	reserved	BOR1MIS	reserved
Type	RO	RO	RO	RO	RW1C	RW1C	RO	RW1C	RW1C	RW1C	RO	RO	RO	RO	RW1C	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.00	Software should not rely on the value of 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	BOR0MIS	RW1C	0	VDD under BOR0 Masked Interrupt Status Value Description 0 When read, a 0 indicates that a BOR0 condition has not occurred. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled because of a BOR0 condition. Writing a 1 to this bit clears it and also the `BOR0RIS` bit in the **RIS** register.
10	VDDAMIS	RW1C	0	VDDA Power OK Masked Interrupt Status Value Description 0 When read, a 0 indicates that VDDA power is good. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled because VDDA was below the proper functioning voltage. Writing a 1 to this bit clears it and also the `VDDARIS` bit in the **RIS** register.
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	MOSCPUPMIS	RW1C	0	MOSC Power Up Masked Interrupt Status Value Description 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. 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 `MOSCPUPRIS` bit in the **RIS** register.
7	USBPLLLMIS	RW1C	0	USB PLL Lock Masked Interrupt Status Value Description 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. 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 `USBPLLLRIS` bit in the **RIS** register.
6	PLLLMIS	RW1C	0	PLL Lock Masked Interrupt Status Value Description 0 When read, a 0 indicates that sufficient time has not passed for the PLL to lock. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the PLL to lock. Writing a 1 to this bit clears it and also the `PLLLRIS` bit in the **RIS** register.
5:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MOFMIS	RO	0	Main Oscillator Failure Masked Interrupt Status Value Description 0 When read, a 0 indicates that the main oscillator has not failed. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled because the main oscillator failed. Writing a 1 to this bit clears it and also the `MOFRIS` bit in the **RIS** register.

Bit/Field	Name	Type	Reset	Description
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	BOR1MIS	RW1C	0	VDD under BOR1 Masked Interrupt Status Value Description 0 When read, a 0 indicates that a BOR1 condition has not occurred. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled because of a BOR1 condition. Writing a 1 to this bit clears it and also the `BOR1RIS` bit in the **RIS** register.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

This register is set with the reset cause after reset. The bits in this register are sticky and maintain their state across multiple reset sequences, 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 RW, 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	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										WDT1	SW	WDT0	BOR	POR	EXT
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	MOSCFAIL	RW	-	MOSC Failure Reset Value Description 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. 1 When read, this bit indicates that the MOSC circuit was enabled for clock validation and failed while the `MOSCIM` bit in the **MOSCCTL** register is clear, generating a reset event.
15:6	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	RW	-	Watchdog Timer 1 Reset Value Description 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. 1 When read, this bit indicates that Watchdog Timer 1 timed out and generated a reset.

Bit/Field	Name	Type	Reset	Description
4	SW	RW	-	Software Reset Value Description 0 When read, this bit indicates that a software reset has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it. 1 When read, this bit indicates that a software reset has caused a reset event.
3	WDT0	RW	-	Watchdog Timer 0 Reset Value Description 0 When read, this bit indicates that Watchdog Timer 0 has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it. 1 When read, this bit indicates that Watchdog Timer 0 timed out and generated a reset.
2	BOR	RW	-	Brown-Out Reset Value Description 0 When read, this bit indicates that a brown-out (BOR0 or BOR1) reset has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it. 1 When read, this bit indicates that a brown-out (BOR0 or BOR1) reset has caused a reset event.
1	POR	RW	-	Power-On Reset Value Description 0 When read, this bit indicates that a power-on reset has not generated a reset. Writing a 0 to this bit clears it. 1 When read, this bit indicates that a power-on reset has caused a reset event.
0	EXT	RW	-	External Reset Value Description 0 When read, this bit indicates that an external reset ($\overline{\text{RST}}$ assertion) has not caused a reset event since the previous power-on reset. Writing a 0 to this bit clears it. 1 When read, this bit indicates that an external reset ($\overline{\text{RST}}$ assertion) has caused a reset event.

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

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

Important: Write the **RCC** register prior to writing the **RCC2** register.

Run-Mode Clock Configuration (RCC)

Base 0x400F.E000
Offset 0x060
Type RW, 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	RW	RW	RW	RW	RW	RW	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			MOSCDIS
Type	RO	RO	RW	RO	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RW
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	RW	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 0 The **Run-Mode Clock Gating Control (RCGCn)** registers are used when the microcontroller enters a sleep mode. 1 The **SCGCn** or **DCGCn** registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The **SCGCn** and **DCGCn** registers allow unused peripherals to consume less power when the microcontroller is in a sleep mode. The **RCGCn** registers are always used to control the clocks in Run mode.
26:23	SYSDIV	RW	0xF	System Clock Divisor Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the `BYPASS` bit in this register is configured). See Table 5-4 on page 211 for bit encodings. If the `SYSDIV` value is less than `MINSYSDIV` (see page 394), and the PLL is being used, then the `MINSYSDIV` value is used as the divisor. If the PLL is not being used, the `SYSDIV` value can be less than `MINSYSDIV`.

Bit/Field	Name	Type	Reset	Description
22	USESYSDIV	RW	0	Enable System Clock Divider Value Description 0 The system clock is used undivided. 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. If the `USERCC2` bit in the **RCC2** register is set, then the `SYSDIV2` field in the **RCC2** register is used as the system clock divider rather than the `SYSDIV` field in this register.
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	RW	1	PLL Power Down Value Description 0 The PLL is operating normally. 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.
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	RW	1	PLL Bypass Value Description 0 The system clock is the PLL output clock divided by the divisor specified by `SYSDIV`. 1 The system clock is derived from the OSC source and divided by the divisor specified by `SYSDIV`. See Table 5-4 on page 211 for programming guidelines. **Note:** The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.

Bit/Field	Name	Type	Reset	Description
10:6	XTAL	RW	0x0B	Crystal Value This field specifies the crystal value attached to the main oscillator. The encoding for this field is provided below. 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 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
0x00-0x5	reserved	
0x06	4 MHz	reserved
0x07	4.096 MHz	reserved
0x08	4.9152 MHz	reserved
0x09	5 MHz (USB)	
0x0A	5.12 MHz	
0x0B	6 MHz (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	
0x17	18.0 MHz (USB)	
0x18	20.0 MHz (USB)	
0x19	24.0 MHz (USB)	
0x1A	25.0 MHz (USB)	

Bit/Field	Name	Type	Reset	Description
5:4	OSCSRC	RW	0x1	Oscillator Source Selects the input source for the OSC. The values are: Value Input Source 0x0 MOSC Main oscillator 0x1 PIOSC Precision internal oscillator (default) 0x2 PIOSC/4 Precision internal oscillator / 4 0x3 LFIOSC Low-frequency internal oscillator For additional oscillator sources, see the **RCC2** register.
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	MOSCDIS	RW	1	Main Oscillator Disable Value Description 0 The main oscillator is enabled. 1 The main oscillator is disabled (default).

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

This register controls which internal bus is used to access each GPIO port. When a bit is clear, the corresponding GPIO port is accessed across the legacy Advanced Peripheral Bus (APB) bus and through the APB memory aperture. When a bit is set, the corresponding port is accessed across the Advanced High-Performance Bus (AHB) bus and through the AHB memory aperture. Each GPIO port can be individually configured to use AHB or APB, but may be accessed only through one aperture. The AHB bus provides better back-to-back access performance than the APB bus. The address aperture in the memory map changes for the ports that are enabled for AHB access (see Table 9-6 on page 592).

Important: Ports K-N and P-Q are only available on the AHB bus, and therefore the corresponding bits reset to 1. If one of these bits is cleared, the corresponding port is disabled. If any of these ports is in use, read-modify-write operations should be used to change the value of this register so that these ports remain enabled.

GPIO High-Performance Bus Control (GPIOHBCTL)

Base 0x400F.E000
Offset 0x06C
Type RW, reset 0x0000.7E00

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									PORTG	PORTF	PORTE	PORTD	PORTC	PORTB	PORTA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	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.
6	PORTG	RW	0	Port G Advanced High-Performance Bus This bit defines the memory aperture for Port G. Value Description 0 Advanced Peripheral Bus (APB). This bus is the legacy bus. 1 Advanced High-Performance Bus (AHB)
5	PORTF	RW	0	Port F Advanced High-Performance Bus This bit defines the memory aperture for Port F. Value Description 0 Advanced Peripheral Bus (APB). This bus is the legacy bus. 1 Advanced High-Performance Bus (AHB)

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

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

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

Table 5-8. RCC2 Fields that Override RCC Fields

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

Important: Write the **RCC** register prior to writing the **RCC2** register.

Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type RW, reset 0x07C0.6810

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	USBPWRDN	PWRDN2	reserved	BYPASS2	reserved				OSCSRC2			reserved			
Type	RO	RW	RW	RO	RW	RO	RO	RO	RO	RW	RW	RW	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	RW	0	Use **RCC2**

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

Bit/Field	Name	Type	Reset	Description
30	DIV400	RW	0	Divide PLL as 400 MHz versus 200 MHz

This bit, along with the SYSDIV2LSB bit, allows additional frequency choices.

Value	Description
0	Use SYSDIV2 as is and apply to 200 MHz predivided PLL output. See Table 5-5 on page 212 for programming guidelines.
1	Append the SYSDIV2LSB bit to the SYSDIV2 field to create a 7 bit divisor using the 400 MHz PLL output, see Table 5-6 on page 212.

Bit/Field	Name	Type	Reset	Description
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	RW	0x0F	System Clock Divisor 2 Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the `BYPASS2` bit is configured). `SYSDIV2` is used for the divisor when both the `USESYSDIV` bit in the **RCC** register and the `USERCC2` bit in this register are set. See Table 5-5 on page 212 for programming guidelines.
22	SYSDIV2LSB	RW	1	Additional LSB for `SYSDIV2` When `DIV400` is set, this bit becomes the LSB of `SYSDIV2`. If `DIV400` is clear, this bit is not used. See Table 5-5 on page 212 for programming guidelines. This bit can only be set or cleared when `DIV400` is set.
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	RW	1	Power-Down USB PLL Value Description 0 The USB PLL operates normally. 1 The USB PLL is powered down.
13	PWRDN2	RW	1	Power-Down PLL 2 Value Description 0 The PLL operates normally. 1 The PLL is powered down.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	BYPASS2	RW	1	PLL Bypass 2 Value Description 0 The system clock is the PLL output clock divided by the divisor specified by `SYSDIV2`. 1 The system clock is derived from the OSC source and divided by the divisor specified by `SYSDIV2`. See Table 5-5 on page 212 for programming guidelines. **Note:** The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.
10:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
6:4	OSCSRC2	RW	0x1	Oscillator Source 2 Selects the input source for the OSC. The values are: Value — Description 0x0 — MOSC Main oscillator 0x1 — PIOSC Precision internal oscillator 0x2 — PIOSC/4 Precision internal oscillator / 4 0x3 — LFIOSC Low-frequency internal oscillator 0x4-0x7 — Reserved
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

This register provides control over the features of the main oscillator, including the ability to enable the MOSC clock verification circuit, what action to take when the MOSC fails, and whether or not a crystal is connected. When enabled, this circuit monitors the frequency of the MOSC to verify that the oscillator is operating within specified limits. If the clock goes invalid after being enabled, the microcontroller issues a power-on reset and reboots to the NMI handler or generates an interrupt.

Main Oscillator Control (MOSCCTL)

Base 0x400F.E000
Offset 0x07C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													NOXTAL	MOSCIM	CVAL
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	NOXTAL	RW	0	No Crystal Connected

Value	Description
0	This bit should be cleared when a crystal or oscillator is connected to the `OSC0` and `OSC1` inputs, regardless of whether or not the MOSC is used or powered down.
1	This bit should be set when a crystal or external oscillator is not connected to the `OSC0` and `OSC1` inputs to reduce power consumption.

Bit/Field	Name	Type	Reset	Description
1	MOSCIM	RW	0	MOSC Failure Action

Value	Description
0	If the MOSC fails, a MOSC failure reset is generated and reboots to the NMI handler.
1	If the MOSC fails, an interrupt is generated as indicated by the `MOFRIS` bit in the **RIS** register..

Regardless of the action taken, if the MOSC fails, the oscillator source is switched to the PIOSC automatically.

Bit/Field	Name	Type	Reset	Description
0	CVAL	RW	0	Clock Validation for MOSC

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

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

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

Deep Sleep Clock Configuration (DSLPCLKCFG)

Base 0x400F.E000
Offset 0x144
Type RW, 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	RW	RW	RW	RW	RW	RW	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		PIOSCPD	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	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	RW	0x0F	Divider Field Override 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. Value Description 0x0 /1 0x1 /2 0x2 /3 0x3 /4 ... ... 0x3F /64
22:7	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
6:4	DSOSCSRC	RW	0x0	Clock Source Specifies the clock source during Deep-Sleep mode. Value Description 0x0 MOSC Use the main oscillator as the source. To use the MOSC as the Deep-Sleep mode clock source, the MOSC must also be configured as the Run mode clock source in the **Run-Mode Clock Configuration (RCC)** register. **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. 0x1 PIOSC Use the precision internal 16-MHz oscillator as the source. 0x2 Reserved 0x3 LFIOSC Use the low-frequency internal oscillator as the source. 0x4-0x7 Reserved
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PIOSCPD	RW	0	PIOSC Power Down Request Allows software to request the PIOSC to be powered-down in Deep-Sleep mode. If the PIOSC is needed by an enabled peripheral during Deep-Sleep, the PIOSC is powered down, but a warning is generated using the `PPDW` bit in the **SDPMST** register. If it is not possible to power down the PIOSC, an error is reported using the `PPDERR` bit in the **SDPMST** register. This bit can only be used to power down the PIOSC when the `PIOSCPDE` bit in the **SYSPROP** register is set. Value Description 0 No action. 1 Software requests that the PIOSC is powered down during Deep-Sleep mode.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 13: System Properties (SYSPROP), offset 0x14C

This register provides information on whether certain System Control properties are present on the microcontroller.

System Properties (SYSPROP)

Base 0x400F.E000
Offset 0x14C
Type RO, reset 0x0000.1D31

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			PIOSCPDE	SRAMSM	SRAMLPM	reserved	FLASHLPM	reserved		reserved		reserved			FPU
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	PIOSCPDE	RO	0x1	PIOSC Power Down Present This bit determines whether the `PIOSCPD` bit in the **DSLPCLKCFG** register can be set to power down the PIOSC in Deep-Sleep mode. Value Description 0 The status of the `PIOSCPD` bit is ignored. 1 The `PIOSCPD` bit can be set to power down the PIOSC in Deep-Sleep mode.
11	SRAMSM	RO	0x1	SRAM Sleep/Deep-Sleep Standby Mode Present This bit determines whether the `SRAMPM` field in the **SLPPWRCFG** and **DSLPPWRCFG** registers can be configured to put the SRAM into Standby mode while in Sleep or Deep-Sleep mode. Value Description 0 A value of 0x1 in the `SRAMPM` fields is ignored. 1 The `SRAMPM` fields can be configured to put the SRAM into Standby mode while in Sleep or Deep-Sleep mode.
10	SRAMLPM	RO	0x1	SRAM Sleep/Deep-Sleep Low Power Mode Present This bit determines whether the `SRAMPM` field in the **SLPPWRCFG** and **DSLPPWRCFG** registers can be configured to put the SRAM into Low Power mode while in Sleep or Deep-Sleep mode. Value Description 0 A value of 0x3 in the `SRAMPM` fields is ignored. 1 The `SRAMPM` fields can be configured to put the SRAM into Low Power mode while in Sleep or Deep-Sleep mode.

Bit/Field	Name	Type	Reset	Description
9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	FLASHLPM	RO	0x1	Flash Memory Sleep/Deep-Sleep Low Power Mode Present This bit determines whether the FLASHPM field in the **SLPPWRCFG** and **DSLPPWRCFG** registers can be configured to put the Flash memory into Low Power mode while in Sleep or Deep-Sleep mode. Value Description 0 A value of 0x2 in the FLASHPM fields is ignored. 1 The FLASHPM fields can be configured to put the Flash memory into Low Power mode while in Sleep or Deep-Sleep mode.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	reserved	RO	0x3	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FPU	RO	0x1	FPU Present This bit indicates if the FPU is present in the Cortex-M4 core. Value Description 0 FPU is not present. 1 FPU is present.

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

This register provides the ability to update or recalibrate the precision internal oscillator.

Precision Internal Oscillator Calibration (PIOSCCAL)

Base 0x400F.E000
Offset 0x150
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	UTEN	reserved														
Type	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							UPDATE	reserved	UT						
Type	RO	RO	RO	RO	RO	RO	RO	RW	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	UTEN	RW	0	Use User Trim Value Value Description 0 The factory calibration value is used for an update trim operation. 1 The trim value in bits[6:0] of this register are used for any update trim operation.
30:9	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.
8	UPDATE	RW	0	Update Trim Value Description 0 No action. 1 Updates the PIOSC trim value with the UT bit. Used with UTEN. 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.
6:0	UT	RW	0x0	User Trim Value User trim value that can be loaded into the PIOSC. Refer to “Precision Internal Oscillator Operation (PIOSC)” on page 212 for more information on calibrating the PIOSC.

Register 15: PLL Frequency 0 (PLLFREQ0), offset 0x160

This register always contains the current M value presented to the system PLL.

The PLL frequency can be calculated using the following equation:

```
PLL frequency = (XTAL frequency * MDIV) / ((Q + 1) * (N + 1))
```

where

```
MDIV = MINT + (MFRAC / 1024)
```

The Q and N values are shown in the **PLLFREQ1** register. Table 21-14 on page 1138 shows the M, Q, and N values as well as the resulting PLL frequency for the various XTAL configurations.

PLL Frequency 0 (PLLFREQ0)

Base 0x400F.E000
Offset 0x160
Type RO, reset 0x0000.0032

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												MFRAC			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MFRAC						MINT									
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	0	0	1	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.
19:10	MFRAC	RO	0x32	PLL M Fractional Value This field contains the integer value of the PLL M value.
9:0	MINT	RO	0x00	PLL M Integer Value This field contains the integer value of the PLL M value.

Register 16: PLL Frequency 1 (PLLFREQ1), offset 0x164

This register always contains the current Q and N values presented to the system PLL.

The M value is shown in the **PLLFREQ0** register. Table 21-14 on page 1138 shows the M, Q, and N values as well as the resulting PLL frequency for the various XTAL configurations.

PLL Frequency 1 (PLLFREQ1)

Base 0x400F.E000
Offset 0x164
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			Q					reserved			N				
Type	RO	RO	RO	RO	RO	RO	RO	RO	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
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12:8	Q	RO	0x0	PLL Q Value This field contains the PLL Q value.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	N	RO	0x1	PLL N Value This field contains the PLL N value.

Register 17: PLL Status (PLLSTAT), offset 0x168

This register shows the direct status of the PLL lock.

PLL Status (PLLSTAT)

Base 0x400F.E000
Offset 0x168
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															LOCK
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	LOCK	RO	0x0	PLL Lock

Value	Description
0	The PLL is unpowered or is not yet locked.
1	The PLL is powered and locked.

Register 18: Sleep Power Configuration (SLPPWRCFG), offset 0x188

This register provides configuration information for the power control of the SRAM and Flash memory while in Sleep mode.

Sleep Power Configuration (SLPPWRCFG)

Base 0x400F.E000
Offset 0x188
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										FLASHPM		reserved		SRAMPM	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	FLASHPM	RW	0x0	Flash Power Modes Value Description 0x0 Active Mode Flash memory is not placed in a lower power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in Sleep mode. 0x1 Reserved 0x2 Low Power Mode Flash memory is placed in low power mode. This mode provides the lowers power consumption but requires more time to come out of Sleep mode. 0x3 Reserved
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
1:0	SRAMPM	RW	0x0	SRAM Power Modes This field controls the low power modes of the on-chip SRAM , including the USB SRAM while the microcontroller is in Deep-Sleep mode. Value Description 0x0 Active Mode SRAM is not placed in a lower power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in Sleep mode. 0x1 Standby Mode SRAM is place in standby mode while in Sleep mode. 0x2 Reserved 0x3 Low Power Mode SRAM is placed in low power mode. This mode provides the slowest time to sleep and wakeup but the lowest power consumption while in Sleep mode.

Register 19: Deep-Sleep Power Configuration (DSLPPWRCFG), offset 0x18C

This register provides configuration information for the power control of the SRAM and Flash memory while in Deep-Sleep mode.

Deep-Sleep Power Configuration (DSLPPWRCFG)

Base 0x400F.E000
Offset 0x18C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										FLASHPM		reserved		SRAMPM	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	FLASHPM	RW	0x0	Flash Power Modes

Value	Description
0x0	Active Mode Flash memory is not placed in a lower power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in Deep-Sleep mode.
0x1	Reserved
0x2	Low Power Mode Flash memory is placed in low power mode. This mode provides the lowers power consumption but requires more time to come out of Deep-Sleep mode.
0x3	Reserved

Bit/Field	Name	Type	Reset	Description
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
1:0	SRAMPM	RW	0x0	SRAM Power Modes This field controls the low power modes of the on-chip SRAM , including the USB SRAM while the microcontroller is in Deep-Sleep mode.

Value	Description
0x0	Active Mode SRAM is not placed in a lower power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in Deep-Sleep mode.
0x1	Standby Mode SRAM is place in standby mode while in Deep-Sleep mode.
0x2	Reserved
0x3	Low Power Mode SRAM is placed in low power mode. This mode provides the slowest time to sleep and wakeup but the lowest power consumption while in Deep-Sleep mode.

Register 20: LDO Sleep Power Control (LDOSPCTL), offset 0x1B4

This register specifies the LDO output voltage while in Sleep mode. Writes to the VLDO bit field have no effect on the LDO output voltage, regardless of what is specified for the VADJEN bit. The LDO output voltage is fixed at the recommended factory reset value.

The table below shows the maximum system clock frequency and PIOSC frequency with respect to the configured LDO voltage.

Operating Voltage (LDO)	Maximum System Clock Frequency	PIOSC
1.2	80 MHz	16 MHz
0.9	20 MHz	16 MHz

Note:
- The LDO will not automatically adjust in Sleep/Deepsleep mode if a debugger has been connected since the last power-on reset.
- If the LDO voltage is adjusted, it will take an extra 4 us to wake up from Sleep or Deep-Sleep mode.

LDO Sleep Power Control (LDOSPCTL)

Base 0x400F.E000
Offset 0x1B4
Type RW, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	VADJEN	reserved														
Type	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								VLDO							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31	VADJEN	RW	0	Voltage Adjust Enable This bit enables the value of the VLDO field to be used to specify the output voltage of the LDO in Sleep mode. Value Description 0 The LDO output voltage is set to the factory default value in Sleep mode. The value of the VLDO field does not affect the LDO operation. 1 The LDO output value in Sleep mode is configured by the value in the VLDO field.
30:8	reserved	RO	0x000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:0	VLDO	RW	0x18	LDO Output Voltage This field provides program control of the LDO output voltage in Run mode. The value of the field is only used for the LDO voltage when the `VADJEN` bit is set. For lowest power in Sleep mode, it is recommended to configure an LDO output voltage that is equal to or lower than the default value of 1.2 V.

Value	Description
0x12	0.90 V
0x13	0.95 V
0x14	1.00 V
0x15	1.05 V
0x16	1.10 V
0x17	1.15 V
0x18	1.20 V
0x19 - 0xFF	reserved

Register 21: LDO Sleep Power Calibration (LDOSPCAL), offset 0x1B8

This register provides factory determined values that are recommended for the VLDO field in the **LDOSPCTL** register while in Sleep mode.

LDO Sleep Power Calibration (LDOSPCAL)

Base 0x400F.E000
Offset 0x1B8
Type RO, reset 0x0000.1818

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WITHPLL								NOPLL							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	WITHPLL	RO	0x18	Sleep with PLL The value in this field is the suggested value for the VLDO field in the **LDOSPCTL** register when using the PLL. This value provides the lowest recommended LDO output voltage for use with the PLL at the maximum specified value.
7:0	NOPLL	RO	0x18	Sleep without PLL The value in this field is the suggested value for the VLDO field in the **LDOSPCTL** register when not using the PLL. This value provides the lowest recommended LDO output voltage for use without the PLL.

Register 22: LDO Deep-Sleep Power Control (LDODPCTL), offset 0x1BC

This register specifies the LDO output voltage while in Deep-Sleep mode. This register must be configured in Run mode before entering Deep-Sleep. Writes to the VLDO bit field have no effect on the LDO output voltage, regardless of what is specified for the VADJEN bit. The LDO output voltage is fixed at the recommended factory reset value.

The table below shows the maximum system clock frequency and PIOSC frequency with respect to the configured LDO voltage.

Operating Voltage (LDO)	Maximum System Clock Frequency	PIOSC
1.2	80 MHz	16 MHz
0.9	20 MHz	16 MHz

Note:
- The LDO will not automatically adjust in Sleep/Deepsleep mode if a debugger has been connected since the last power-on reset.
- If the LDO voltage is adjusted, it will take an extra 4 us to wake up from Sleep or Deep-Sleep mode.

LDO Deep-Sleep Power Control (LDODPCTL)

Base 0x400F.E000
Offset 0x1BC
Type RW, reset 0x0000.0012

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	VADJEN	reserved														
Type	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								VLDO							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

Bit/Field	Name	Type	Reset	Description
31	VADJEN	RW	0	Voltage Adjust Enable This bit enables the value of the VLDO field to be used to specify the output voltage of the LDO in Deep-Sleep mode. Value Description 0 The LDO output voltage is set to the factory default value in Deep-Sleep mode. The value of the VLDO field does not affect the LDO operation. 1 The LDO output value in Deep-Sleep mode is configured by the value in the VLDO field.
30:8	reserved	RO	0x000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:0	VLDO	RW	0x12	LDO Output Voltage This field provides program control of the LDO output voltage in Run mode. The value of the field is only used for the LDO voltage when the VADJEN bit is set. For lowest power in Deep-Sleep mode, it is recommended to configure the LDO output voltage to the default value of 0.90 V.

Value	Description
0x12	0.90 V
0x13	0.95 V
0x14	1.00 V
0x15	1.05 V
0x16	1.10 V
0x17	1.15 V
0x18	1.20 V
0x19 - 0xFF	reserved

Register 23: LDO Deep-Sleep Power Calibration (LDODPCAL), offset 0x1C0

This register provides factory determined values that are recommended for the VLDO field in the **LDODPCTL** register while in Deep-Sleep mode.

LDO Deep-Sleep Power Calibration (LDODPCAL)

Base 0x400F.E000
Offset 0x1C0
Type RO, reset 0x0000.1212

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NOPLL								30KHZ							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	NOPLL	RO	0x12	Deep-Sleep without PLL The value in this field is the suggested value for the VLDO field in the **LDODPCTL** register when not using the PLL. This value provides the lowest recommended LDO output voltage for use with the system clock.
7:0	30KHZ	RO	0x12	Deep-Sleep with IOSC The value in this field is the suggested value for the VLDO field in the **LDODPCTL** register when not using the PLL. This value provides the lowest recommended LDO output voltage for use with the low-frequency internal oscillator.

Register 24: Sleep / Deep-Sleep Power Mode Status (SDPMST), offset 0x1CC

This register provides status information on the Sleep and Deep-Sleep power modes as well as some real time status that can be viewed by a debugger or the core if it is running. These events do not trigger an interrupt and are meant to provide information that can help tune software for power management. The status register gets written at the beginning of every Dynamic Power Management event request with the results of any error checking. There is no mechanism to clear the bits; they are overwritten on the next event. The LDOUA, FLASHLP, LOWPWR, PRACT bits provide real time data and there are no events to register that information.

Sleep / Deep-Sleep Power Mode Status (SDPMST)

Base 0x400F.E000
Offset 0x1CC
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												LDOUA	FLASHLP	LOWPWR	PRACT
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PPDW	LMAXERR	reserved	LSMINERR	LDMINERR	PPDERR	FPDERR	SPDERR
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	LDOUA	RO	0	LDO Update Active Value Description 0 The LDO voltage level is not changing. 1 The LDO voltage level is changing.
18	FLASHLP	RO	0	Flash Memory in Low Power State Value Description 0 The Flash memory is currently in the active state. 1 The Flash memory is currently in the low power state as programmed in the **SLPPWRCFG** or **DSLPPWRCFG** register.
17	LOWPWR	RO	0	Sleep or Deep-Sleep Mode Value Description 0 The microcontroller is currently in Run mode. 1 The microcontroller is currently in Sleep or Deep-Sleep mode and is waiting for an interrupt or is in the process of powering up. The status of this bit is not affected by the power state of the Flash memory or SRAM.

Bit/Field	Name	Type	Reset	Description
16	PRACT	RO	0	Sleep or Deep-Sleep Power Request Active Value Description 0 A power request is not active. 1 The microcontroller is currently in Deep-Sleep mode or is in Sleep mode and a request to put the SRAM and/or Flash memory into a lower power mode is currently active as configured by the **SLPPWRCFG** register.
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PPDW	RO	0	PIOSC Power Down Request Warning Value Description 0 No error. 1 A warning has occurred because software has requested that the PIOSC be powered down during Deep-Sleep using the `PIOSCPD` bit in the **DSLPCLKCFG** register and a peripheral requires that it be active in Deep-Sleep. The PIOSC is powered down regardless of the warning.
6	LMAXERR	RO	0	`VLDO` Value Above Maximum Error Value Description 0 No error. 1 An error has occurred because software has requested that the LDO voltage be above the maximum value allowed using the `VLDO` bit in the **LDOSPCTL** or **LDODPCTL** register. In this situation, the LDO is set to the factory default value.
5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	LSMINERR	RO	0	`VLDO` Value Below Minimum Error in Sleep Mode Value Description 0 No error. 1 An error has occurred because software has requested that the LDO voltage be below the minimum value allowed using the `VLDO` bit in the **LDOSPCTL** register. In this situation, the LDO voltage is not changed when entering Sleep mode.

Bit/Field	Name	Type	Reset	Description
3	LDMINERR	RO	0	`VLDO` Value Below Minimum Error in Deep-Sleep Mode Value Description 0 No error. 1 An error has occurred because software has requested that the LDO voltage be below the minimum value allowed using the `VLDO` bit in the **LDODPCTL** register. In this situation, the LDO voltage is not changed when entering Deep-Sleep mode.
2	PPDERR	RO	0	PIOSC Power Down Request Error Value Description 0 No error. 1 An error has occurred because software has requested that the PIOSC be powered down during Deep-Sleep and it is not possible to power down the PIOSC. In this situation, the PIOSC is not powered down when entering Deep-Sleep mode.
1	FPDERR	RO	0	Flash Memory Power Down Request Error Value Description 0 No error. 1 An error has occurred because software has requested a Flash memory power down mode that is not available using the `FLASHPM` field in the **SLPPWRCFG** or the **DSLPPWRCFG** register.
0	SPDERR	RO	0	SRAM Power Down Request Error Value Description 0 No error. 1 An error has occurred because software has requested an SRAM power down mode that is not available using the `SRAMPM` field in the **SLPPWRCFG** or the **DSLPPWRCFG** register.

Register 25: Watchdog Timer Peripheral Present (PPWD), offset 0x300

The **PPWD** register provides software information regarding the watchdog modules.

Important: This register should be used to determine which watchdog timers are implemented on this microcontroller. However, to support legacy software, the **DC1** register is available. A read of the **DC1** register correctly identifies if a legacy module is present.

Watchdog Timer Peripheral Present (PPWD)

Base 0x400F.E000
Offset 0x300
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														P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	P1	RO	0x1	Watchdog Timer 1 Present Value Description 0 Watchdog module 1 is not present. 1 Watchdog module 1 is present.
0	P0	RO	0x1	Watchdog Timer 0 Present Value Description 0 Watchdog module 0 is not present. 1 Watchdog module 0 is present.

Register 26: 16/32-Bit General-Purpose Timer Peripheral Present (PPTIMER), offset 0x304

The **PPTIMER** register provides software information regarding the 16/32-bit general-purpose timer modules.

Important: This register should be used to determine which timers are implemented on this microcontroller. However, to support legacy software, the **DC2** register is available. A read of the **DC2** register correctly identifies if a legacy module is present. Software must use this register to determine if a module that is not supported by the **DC2** register is present.

16/32-Bit General-Purpose Timer Peripheral Present (PPTIMER)

Base 0x400F.E000
Offset 0x304
Type RO, 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										P5	P4	P3	P2	P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	P5	RO	0x1	16/32-Bit General-Purpose Timer 5 Present Value Description 0 16/32-bit general-purpose timer module 6 is not present. 1 16/32-bit general-purpose timer module 5 is present.
4	P4	RO	0x1	16/32-Bit General-Purpose Timer 4 Present Value Description 0 16/32-bit general-purpose timer module 4 is not present. 1 16/32-bit general-purpose timer module 4 is present.
3	P3	RO	0x1	16/32-Bit General-Purpose Timer 3 Present Value Description 0 16/32-bit general-purpose timer module 3 is not present. 1 16/32-bit general-purpose timer module 3 is present.

Bit/Field	Name	Type	Reset	Description
2	P2	RO	0x1	16/32-Bit General-Purpose Timer 2 Present Value Description 0 16/32-bit general-purpose timer module 2 is not present. 1 16/32-bit general-purpose timer module 2 is present.
1	P1	RO	0x1	16/32-Bit General-Purpose Timer 1 Present Value Description 0 16/32-bit general-purpose timer module 1 is not present. 1 16/32-bit general-purpose timer module 1 is present.
0	P0	RO	0x1	16/32-Bit General-Purpose Timer 0 Present Value Description 0 16/32-bit general-purpose timer module 0 is not present. 1 16/32-bit general-purpose timer module 0 is present.

Register 27: General-Purpose Input/Output Peripheral Present (PPGPIO), offset 0x308

The **PPGPIO** register provides software information regarding the general-purpose input/output modules.

Important: This register should be used to determine which GPIO ports are implemented on this microcontroller. However, to support legacy software, the **DC4** register is available. A read of the **DC4** register correctly identifies if a legacy module is present. Software must use this register to determine if a module that is not supported by the **DC4** register is present.

General-Purpose Input/Output Peripheral Present (PPGPIO)

Base 0x400F.E000
Offset 0x308
Type RO, 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	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:15	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	P14	RO	0x0	GPIO Port Q Present Value Description 0 GPIO Port Q is not present. 1 GPIO Port Q is present.
13	P13	RO	0x0	GPIO Port P Present Value Description 0 GPIO Port P is not present. 1 GPIO Port P is present.
12	P12	RO	0x0	GPIO Port N Present Value Description 0 GPIO Port N is not present. 1 GPIO Port N is present.

Bit/Field	Name	Type	Reset	Description
11	P11	RO	0x0	GPIO Port M Present Value Description 0 GPIO Port M is not present. 1 GPIO Port M is present.
10	P10	RO	0x0	GPIO Port L Present Value Description 0 GPIO Port L is not present. 1 GPIO Port L is present.
9	P9	RO	0x0	GPIO Port K Present Value Description 0 GPIO Port K is not present. 1 GPIO Port K is present.
8	P8	RO	0x0	GPIO Port J Present Value Description 0 GPIO Port J is not present. 1 GPIO Port J is present.
7	P7	RO	0x0	GPIO Port H Present Value Description 0 GPIO Port H is not present. 1 GPIO Port H is present.
6	P6	RO	0x1	GPIO Port G Present Value Description 0 GPIO Port G is not present. 1 GPIO Port G is present.
5	P5	RO	0x1	GPIO Port F Present Value Description 0 GPIO Port F is not present. 1 GPIO Port F is present.

Bit/Field	Name	Type	Reset	Description
4	P4	RO	0x1	GPIO Port E Present Value Description 0 GPIO Port E is not present. 1 GPIO Port E is present.
3	P3	RO	0x1	GPIO Port D Present Value Description 0 GPIO Port D is not present. 1 GPIO Port D is present.
2	P2	RO	0x1	GPIO Port C Present Value Description 0 GPIO Port C is not present. 1 GPIO Port C is present.
1	P1	RO	0x1	GPIO Port B Present Value Description 0 GPIO Port B is not present. 1 GPIO Port B is present.
0	P0	RO	0x1	GPIO Port A Present Value Description 0 GPIO Port A is not present. 1 GPIO Port A is present.

Register 28: Micro Direct Memory Access Peripheral Present (PPDMA), offset 0x30C

The **PPDMA** register provides software information regarding the µDMA module.

Important: This register should be used to determine if the µDMA module is implemented on this microcontroller. However, to support legacy software, the **DC7** register is available. A read of the **DC7** register correctly identifies if the µDMA module is present.

Micro Direct Memory Access Peripheral Present (PPDMA)

Base 0x400F.E000
Offset 0x30C
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															P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	P0	RO	0x1	µDMA Module Present Value Description 0 µDMA module is not present. 1 µDMA module is present.

Register 29: Hibernation Peripheral Present (PPHIB), offset 0x314

The **PPHIB** register provides software information regarding the Hibernation module.

Important: This register should be used to determine if the Hibernation module is implemented on this microcontroller. However, to support legacy software, the **DC1** register is available. A read of the **DC1** register correctly identifies if the Hibernation module is present.

Hibernation Peripheral Present (PPHIB)

Base 0x400F.E000
Offset 0x314
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															P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	P0	RO	0x0	Hibernation Module Present Value Description 0 Hibernation module is not present. 1 Hibernation module is present.

Register 30: Universal Asynchronous Receiver/Transmitter Peripheral Present (PPUART), offset 0x318

The **PPUART** register provides software information regarding the UART modules.

Important: This register should be used to determine which UART modules are implemented on this microcontroller. However, to support legacy software, the **DC2** register is available. A read of the **DC2** register correctly identifies if a legacy UART module is present. Software must use this register to determine if a module that is not supported by the **DC2** register is present.

Universal Asynchronous Receiver/Transmitter Peripheral Present (PPUART)

Base 0x400F.E000
Offset 0x318
Type RO, reset 0x0000.00FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								P7	P6	P5	P4	P3	P2	P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	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
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	P7	RO	0x1	UART Module 7 Present Value Description 0 UART module 7 is not present. 1 UART module 7 is present.
6	P6	RO	0x1	UART Module 6 Present Value Description 0 UART module 6 is not present. 1 UART module 6 is present.
5	P5	RO	0x1	UART Module 5 Present Value Description 0 UART module 5 is not present. 1 UART module 5 is present.

Bit/Field	Name	Type	Reset	Description
4	P4	RO	0x1	UART Module 4 Present Value Description 0 UART module 4 is not present. 1 UART module 4 is present.
3	P3	RO	0x1	UART Module 3 Present Value Description 0 UART module 3 is not present. 1 UART module 3 is present.
2	P2	RO	0x1	UART Module 2 Present Value Description 0 UART module 2 is not present. 1 UART module 2 is present.
1	P1	RO	0x1	UART Module 1 Present Value Description 0 UART module 1 is not present. 1 UART module 1 is present.
0	P0	RO	0x1	UART Module 0 Present Value Description 0 UART module 0 is not present. 1 UART module 0 is present.

Register 31: Synchronous Serial Interface Peripheral Present (PPSSI), offset 0x31C

The **PPSSI** register provides software information regarding the SSI modules.

Important: This register should be used to determine which SSI modules are implemented on this microcontroller. However, to support legacy software, the **DC2** register is available. A read of the **DC2** register correctly identifies if a legacy SSI module is present. Software must use this register to determine if a module that is not supported by the **DC2** register is present.

Synchronous Serial Interface Peripheral Present (PPSSI)

Base 0x400F.E000
Offset 0x31C
Type RO, reset 0x0000.000F

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												P3	P2	P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	P3	RO	0x1	SSI Module 3 Present
2	P2	RO	0x1	SSI Module 2 Present
1	P1	RO	0x1	SSI Module 1 Present

Bit 3, P3:

Value	Description
0	SSI module 3 is not present.
1	SSI module 3 is present.

Bit 2, P2:

Value	Description
0	SSI module 2 is not present.
1	SSI module 2 is present.

Bit 1, P1:

Value	Description
0	SSI module 1 is not present.
1	SSI module 1 is present.

Bit/Field	Name	Type	Reset	Description
0	P0	RO	0x1	SSI Module 0 Present Value Description 0 SSI module 0 is not present. 1 SSI module 0 is present.

Register 32: Inter-Integrated Circuit Peripheral Present (PPI2C), offset 0x320

The **PPI2C** register provides software information regarding the I^{2}C modules.

Important: This register should be used to determine which I^{2}C modules are implemented on this microcontroller. However, to support legacy software, the **DC2** register is available. A read of the **DC2** register correctly identifies if a legacy I^{2}C module is present. Software must use this register to determine if a module that is not supported by the **DC2** register is present.

Inter-Integrated Circuit Peripheral Present (PPI2C)

Base 0x400F.E000
Offset 0x320
Type RO, 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										P5	P4	P3	P2	P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	P5	RO	0x1	I^{2}C Module 5 Present Value Description 0 I^{2}C module 5 is not present. 1 I^{2}C module 5 is present.
4	P4	RO	0x1	I^{2}C Module 4 Present Value Description 0 I^{2}C module 4 is not present. 1 I^{2}C module 4 is present.
3	P3	RO	0x1	I^{2}C Module 3 Present Value Description 0 I^{2}C module 3 is not present. 1 I^{2}C module 3 is present.

Bit/Field	Name	Type	Reset	Description
2	P2	RO	0x1	I^{2}C Module 2 Present Value Description 0 I^{2}C module 2 is not present. 1 I^{2}C module 2 is present.
1	P1	RO	0x1	I^{2}C Module 1 Present Value Description 0 I^{2}C module 1 is not present. 1 I^{2}C module 1 is present.
0	P0	RO	0x1	I^{2}C Module 0 Present Value Description 0 I^{2}C module 0 is not present. 1 I^{2}C module 0 is present.

Register 33: Universal Serial Bus Peripheral Present (PPUSB), offset 0x328

The **PPUSB** register provides software information regarding the USB module.

Important: This register should be used to determine if the USB module is implemented on this microcontroller. However, to support legacy software, the **DC6** register is available. A read of the **DC6** register correctly identifies if the USB module is present.

Universal Serial Bus Peripheral Present (PPUSB)

Base 0x400F.E000
Offset 0x328
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															P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	P0	RO	0x1	USB Module Present

Value	Description
0	USB module is not present.
1	USB module is present.

Register 34: Controller Area Network Peripheral Present (PPCAN), offset 0x334

The **PPCAN** register provides software information regarding the CAN modules.

Important: This register should be used to determine which CAN modules are implemented on this microcontroller. However, to support legacy software, the **DC1** register is available. A read of the **DC1** register correctly identifies if a legacy CAN module is present.

Controller Area Network Peripheral Present (PPCAN)

Base 0x400F.E000
Offset 0x334
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														P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	P1	RO	0x0	CAN Module 1 Present Value Description 0 CAN module 1 is not present. 1 CAN module 1 is present.
0	P0	RO	0x1	CAN Module 0 Present Value Description 0 CAN module 0 is not present. 1 CAN module 0 is present.

Register 35: Analog-to-Digital Converter Peripheral Present (PPADC), offset 0x338

The **PPADC** register provides software information regarding the ADC modules.

Important: This register should be used to determine which ADC modules are implemented on this microcontroller. However, to support legacy software, the **DC1** register is available. A read of the **DC1** register correctly identifies if a legacy ADC module is present.

Analog-to-Digital Converter Peripheral Present (PPADC)

Base 0x400F.E000
Offset 0x338
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														P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	P1	RO	0x1	ADC Module 1 Present Value Description 0 ADC module 1 is not present. 1 ADC module 1 is present.
0	P0	RO	0x1	ADC Module 0 Present Value Description 0 ADC module 0 is not present. 1 ADC module 0 is present.

Register 36: Analog Comparator Peripheral Present (PPACMP), offset 0x33C

The **PPACMP** register provides software information regarding the analog comparator module.

Important: This register should be used to determine if the analog comparator module is implemented on this microcontroller. However, to support legacy software, the **DC2** register is available. A read of the **DC2** register correctly identifies if the analog comparator module is present.

Note that the **Analog Comparator Peripheral Properties (ACMPPP)** register indicates how many analog comparator blocks are included in the module.

Analog Comparator Peripheral Present (PPACMP)

Base 0x400F.E000
Offset 0x33C
Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	P0	RO	0x1	Analog Comparator Module Present Value Description 0 Analog comparator module is not present. 1 Analog comparator module is present.

Register 37: Pulse Width Modulator Peripheral Present (PPPWM), offset 0x340

The **PPPWM** register provides software information regarding the PWM modules.

Important: This register should be used to determine which PWM modules are implemented on this microcontroller. However, to support legacy software, the **DC1** register is available. A read of the **DC1** register correctly identifies if the legacy PWM module is present. Software must use this register to determine if a module that is not supported by the **DC1** register is present.

Pulse Width Modulator Peripheral Present (PPPWM)

Base 0x400F.E000
Offset 0x340
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														P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	P1	RO	0x0	PWM Module 1 Present Value Description 0 PWM module 1 is not present. 1 PWM module 1 is present.
0	P0	RO	0x0	PWM Module 0 Present Value Description 0 PWM module 0 is not present. 1 PWM module 0 is present.

Register 38: Quadrature Encoder Interface Peripheral Present (PPQEI), offset 0x344

The **PPQEI** register provides software information regarding the QEI modules.

Important: This register should be used to determine which QEI modules are implemented on this microcontroller. However, to support legacy software, the **DC2** register is available. A read of the **DC2** register correctly identifies if a legacy QEI module is present.

Quadrature Encoder Interface Peripheral Present (PPQEI)

Base 0x400F.E000
Offset 0x344
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														P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	P1	RO	0x0	QEI Module 1 Present Value Description 0 QEI module 1 is not present. 1 QEI module 1 is present.
0	P0	RO	0x0	QEI Module 0 Present Value Description 0 QEI module 0 is not present. 1 QEI module 0 is present.

Register 39: EEPROM Peripheral Present (PPEEPROM), offset 0x358

The **PPEEPROM** register provides software information regarding the EEPROM module.

EEPROM Peripheral Present (PPEEPROM)

Base 0x400F.E000
Offset 0x358
Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	P0	RO	0x1	EEPROM Module Present Value Description 0 EEPROM module is not present. 1 EEPROM module is present.

Register 40: 32/64-Bit Wide General-Purpose Timer Peripheral Present (PPWTIMER), offset 0x35C

The **PPWTIMER** register provides software information regarding the 32/64-bit wide general-purpose timer modules.

32/64-Bit Wide General-Purpose Timer Peripheral Present (PPWTIMER)

Base 0x400F.E000
Offset 0x35C
Type RO, reset 0x0000.003F

	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										P5	P4	P3	P2	P1	P0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	P5	RO	0x1	32/64-Bit Wide General-Purpose Timer 5 Present Value Description 0 32/64-bit wide general-purpose timer module 5 is not present. 1 32/64-bit wide general-purpose timer module 5 is present.
4	P4	RO	0x1	32/64-Bit Wide General-Purpose Timer 4 Present Value Description 0 32/64-bit wide general-purpose timer module 4 is not present. 1 32/64-bit wide general-purpose timer module 4 is present.
3	P3	RO	0x1	32/64-Bit Wide General-Purpose Timer 3 Present Value Description 0 32/64-bit wide general-purpose timer module 3 is not present. 1 32/64-bit wide general-purpose timer module 3 is present.
2	P2	RO	0x1	32/64-Bit Wide General-Purpose Timer 2 Present Value Description 0 32/64-bit wide general-purpose timer module 2 is not present. 1 32/64-bit wide general-purpose timer module 2 is present.

Bit/Field	Name	Type	Reset	Description
1	P1	RO	0x1	32/64-Bit Wide General-Purpose Timer 1 Present Value Description 0 32/64-bit wide general-purpose timer module 1 is not present. 1 32/64-bit wide general-purpose timer module 1 is present.
0	P0	RO	0x1	32/64-Bit Wide General-Purpose Timer 0 Present Value Description 0 32/64-bit wide general-purpose timer module 0 is not present. 1 32/64-bit wide general-purpose timer module 0 is present.

Register 41: Watchdog Timer Software Reset (SRWD), offset 0x500

The **SRWD** register provides software the capability to reset the available watchdog modules. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SRCRn** bits.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRWD** register. While the **SRWD** bit is 1, the peripheral is held in reset.

2. Software completes the reset process by clearing the **SRWD** bit.

There may be latency from the clearing of the **SRWD** bit to when the peripheral is ready for use. Software can check the corresponding **PRWD** bit to be sure.

Important: This register should be used to reset the watchdog modules. To support legacy software, the **SRCR0** register is available. Setting a bit in the **SRCR0** register also resets the corresponding module. Any bits that are changed by writing to the **SRCR0** register can be read back correctly when reading the **SRCR0** register. If software uses this register to reset a legacy peripheral (such as Watchdog 1), the write causes proper operation, but the value of that bit is not reflected in the **SRCR0** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Watchdog Timer Software Reset (SRWD)

Base 0x400F.E000
Offset 0x500
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	R1	RW	0	Watchdog Timer 1 Software Reset Value Description 0 Watchdog module 1 is not reset. 1 Watchdog module 1 is reset.

Bit/Field	Name	Type	Reset	Description
0	R0	RW	0	Watchdog Timer 0 Software Reset Value Description 0 Watchdog module 0 is not reset. 1 Watchdog module 0 is reset.

Register 42: 16/32-Bit General-Purpose Timer Software Reset (SRTIMER), offset 0x504

The **SRTIMER** register provides software the capability to reset the available 16/32-bit timer modules. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the timer modules and has the same bit polarity as the corresponding **SRCRn** bits.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRTIMER** register. While the **SRTIMER** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRTIMER** bit.

There may be latency from the clearing of the **SRTIMER** bit to when the peripheral is ready for use. Software can check the corresponding **PRTIMER** bit to be sure.

Important: This register should be used to reset the timer modules. To support legacy software, the **SRCR1** register is available. Setting a bit in the **SRCR1** register also resets the corresponding module. Any bits that are changed by writing to the **SRCR1** register can be read back correctly when reading the **SRCR1** register. Software must use this register to reset modules that are not present in the legacy registers. If software uses this register to reset a legacy peripheral (such as Timer 1), the write causes proper operation, but the value of that bit is not reflected in the **SRCR1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

16/32-Bit General-Purpose Timer Software Reset (SRTIMER)

Base 0x400F.E000
Offset 0x504
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	R5	RW	0	16/32-Bit General-Purpose Timer 5 Software Reset Value Description 0 16/32-bit general-purpose timer module 5 is not reset. 1 16/32-bit general-purpose timer module 5 is reset.

Bit/Field	Name	Type	Reset	Description
4	R4	RW	0	16/32-Bit General-Purpose Timer 4 Software Reset Value Description 0 16/32-bit general-purpose timer module 4 is not reset. 1 16/32-bit general-purpose timer module 4 is reset.
3	R3	RW	0	16/32-Bit General-Purpose Timer 3 Software Reset Value Description 0 16/32-bit general-purpose timer module 3 is not reset. 1 16/32-bit general-purpose timer module 3 is reset.
2	R2	RW	0	16/32-Bit General-Purpose Timer 2 Software Reset Value Description 0 16/32-bit general-purpose timer module 2 is not reset. 1 16/32-bit general-purpose timer module 2 is reset.
1	R1	RW	0	16/32-Bit General-Purpose Timer 1 Software Reset Value Description 0 16/32-bit general-purpose timer module 1 is not reset. 1 16/32-bit general-purpose timer module 1 is reset.
0	R0	RW	0	16/32-Bit General-Purpose Timer 0 Software Reset Value Description 0 16/32-bit general-purpose timer module 0 is not reset. 1 16/32-bit general-purpose timer module 0 is reset.

Register 43: General-Purpose Input/Output Software Reset (SRGPIO), offset 0x508

The **SRGPIO** register provides software the capability to reset the available GPIO modules. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the GPIO modules and has the same bit polarity as the corresponding **SRCRn** bits.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRGPIO** register. While the **SRGPIO** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRGPIO** bit.

There may be latency from the clearing of the **SRGPIO** bit to when the peripheral is ready for use. Software can check the corresponding **PRGPIO** bit to be sure.

Important: This register should be used to reset the GPIO modules. To support legacy software, the **SRCR2** register is available. Setting a bit in the **SRCR2** register also resets the corresponding module. Any bits that are changed by writing to the **SRCR2** register can be read back correctly when reading the **SRCR2** register. Software must use this register to reset modules that are not present in the legacy registers. If software uses this register to reset a legacy peripheral (such as GPIO A), the write causes proper operation, but the value of that bit is not reflected in the **SRCR2** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

General-Purpose Input/Output Software Reset (SRGPIO)

Base 0x400F.E000
Offset 0x508
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									R6	R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	R6	RW	0	GPIO Port G Software Reset

Value	Description
0	GPIO Port G is not reset.
1	GPIO Port G is reset.

Bit/Field	Name	Type	Reset	Description
5	R5	RW	0	GPIO Port F Software Reset Value Description 0 GPIO Port F is not reset. 1 GPIO Port F is reset.
4	R4	RW	0	GPIO Port E Software Reset Value Description 0 GPIO Port E is not reset. 1 GPIO Port E is reset.
3	R3	RW	0	GPIO Port D Software Reset Value Description 0 GPIO Port D is not reset. 1 GPIO Port D is reset.
2	R2	RW	0	GPIO Port C Software Reset Value Description 0 GPIO Port C is not reset. 1 GPIO Port C is reset.
1	R1	RW	0	GPIO Port B Software Reset Value Description 0 GPIO Port B is not reset. 1 GPIO Port B is reset.
0	R0	RW	0	GPIO Port A Software Reset Value Description 0 GPIO Port A is not reset. 1 GPIO Port A is reset.

Register 44: Micro Direct Memory Access Software Reset (SRDMA), offset 0x50C

The **SRDMA** register provides software the capability to reset the available μDMA module. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the μDMA module and has the same bit polarity as the corresponding **SRCRn** bits.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRDMA** register. While the **SRDMA** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRDMA** bit.

There may be latency from the clearing of the **SRDMA** bit to when the peripheral is ready for use. Software can check the corresponding **PRDMA** bit to be sure.

Important: This register should be used to reset the μDMA module. To support legacy software, the **SRCR2** register is available. Setting the UDMA bit in the **SRCR2** register also resets the μDMA module. If the UDMA bit is set by writing to the **SRCR2** register, it can be read back correctly when reading the **SRCR2** register. If software uses this register to reset the μDMA module, the write causes proper operation, but the value of the UDMA bit is not reflected in the **SRCR2** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Micro Direct Memory Access Software Reset (SRDMA)

Base 0x400F.E000
Offset 0x50C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RW	0	μDMA Module Software Reset

Value	Description
0	μDMA module is not reset.
1	μDMA module is reset.

Register 45: Universal Asynchronous Receiver/Transmitter Software Reset (SRUART), offset 0x518

The **SRUART** register provides software the capability to reset the available UART modules. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the UART modules and has the same bit polarity as the corresponding **SRCRn** bits.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRUART** register. While the **SRUART** bit is 1, the peripheral is held in reset.

2. Software completes the reset process by clearing the **SRUART** bit.

There may be latency from the clearing of the **SRUART** bit to when the peripheral is ready for use. Software can check the corresponding **PRUART** bit to be sure.

Important: This register should be used to reset the UART modules. To support legacy software, the **SRCR1** register is available. Setting a bit in the **SRCR1** register also resets the corresponding module. Any bits that are changed by writing to the **SRCR1** register can be read back correctly when reading the **SRCR1** register. Software must use this register to reset modules that are not present in the legacy registers. If software uses this register to reset a legacy peripheral (such as UART0), the write causes proper operation, but the value of that bit is not reflected in the **SRCR1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Universal Asynchronous Receiver/Transmitter Software Reset (SRUART)

Base 0x400F.E000
Offset 0x518
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								R7	R6	R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	R7	RW	0	UART Module 7 Software Reset

Value	Description
0	UART module 7 is not reset.
1	UART module 7 is reset.

Bit/Field	Name	Type	Reset	Description
6	R6	RW	0	UART Module 6 Software Reset Value Description 0 UART module 6 is not reset. 1 UART module 6 is reset.
5	R5	RW	0	UART Module 5 Software Reset Value Description 0 UART module 5 is not reset. 1 UART module 5 is reset.
4	R4	RW	0	UART Module 4 Software Reset Value Description 0 UART module 4 is not reset. 1 UART module 4 is reset.
3	R3	RW	0	UART Module 3 Software Reset Value Description 0 UART module 3 is not reset. 1 UART module 3 is reset.
2	R2	RW	0	UART Module 2 Software Reset Value Description 0 UART module 2 is not reset. 1 UART module 2 is reset.
1	R1	RW	0	UART Module 1 Software Reset Value Description 0 UART module 1 is not reset. 1 UART module 1 is reset.
0	R0	RW	0	UART Module 0 Software Reset Value Description 0 UART module 0 is not reset. 1 UART module 0 is reset.

Register 46: Synchronous Serial Interface Software Reset (SRSSI), offset 0x51C

The **SRSSI** register provides software the capability to reset the available SSI modules. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the SSI modules and has the same bit polarity as the corresponding **SRCRn** bits.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRSSI** register. While the **SRSSI** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRSSI** bit.

There may be latency from the clearing of the **SRSSI** bit to when the peripheral is ready for use. Software can check the corresponding **PRSSI** bit to be sure.

Important: This register should be used to reset the SSI modules. To support legacy software, the **SRCR1** register is available. Setting a bit in the **SRCR1** register also resets the corresponding module. Any bits that are changed by writing to the **SRCR1** register can be read back correctly when reading the **SRCR1** register. Software must use this register to reset modules that are not present in the legacy registers. If software uses this register to reset a legacy peripheral (such as SSI0), the write causes proper operation, but the value of that bit is not reflected in the **SRCR1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Synchronous Serial Interface Software Reset (SRSSI)

Base 0x400F.E000
Offset 0x51C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	R3	RW	0	SSI Module 3 Software Reset

Value	Description
0	SSI module 3 is not reset.
1	SSI module 3 is reset.

Bit/Field	Name	Type	Reset	Description
2	R2	RW	0	SSI Module 2 Software Reset Value Description 0 SSI module 2 is not reset. 1 SSI module 2 is reset.
1	R1	RW	0	SSI Module 1 Software Reset Value Description 0 SSI module 1 is not reset. 1 SSI module 1 is reset.
0	R0	RW	0	SSI Module 0 Software Reset Value Description 0 SSI module 0 is not reset. 1 SSI module 0 is reset.

Register 47: Inter-Integrated Circuit Software Reset (SRI2C), offset 0x520

The **SRI2C** register provides software the capability to reset the available I^2C modules. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the I^2C modules and has the same bit polarity as the corresponding **SRCRn** bits.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRI2C** register. While the **SRI2C** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRI2C** bit.

There may be latency from the clearing of the **SRI2C** bit to when the peripheral is ready for use. Software can check the corresponding **PRI2C** bit to be sure.

Important: This register should be used to reset the I^2C modules. To support legacy software, the **SRCR1** register is available. Setting a bit in the **SRCR1** register also resets the corresponding module. Any bits that are changed by writing to the **SRCR1** register can be read back correctly when reading the **SRCR1** register. Software must use this register to reset modules that are not present in the legacy registers. If software uses this register to reset a legacy peripheral (such as I2C0), the write causes proper operation, but the value of that bit is not reflected in the **SRCR1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Inter-Integrated Circuit Software Reset (SRI2C)

Base 0x400F.E000
Offset 0x520
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	R5	RW	0	I^2C Module 5 Software Reset Value Description 0 I^2C module 5 is not reset. 1 I^2C module 5 is reset.

Bit/Field	Name	Type	Reset	Description
4	R4	RW	0	I^{2}C Module 4 Software Reset Value Description 0 I^{2}C module 4 is not reset. 1 I^{2}C module 4 is reset.
3	R3	RW	0	I^{2}C Module 3 Software Reset Value Description 0 I^{2}C module 3 is not reset. 1 I^{2}C module 3 is reset.
2	R2	RW	0	I^{2}C Module 2 Software Reset Value Description 0 I^{2}C module 2 is not reset. 1 I^{2}C module 2 is reset.
1	R1	RW	0	I^{2}C Module 1 Software Reset Value Description 0 I^{2}C module 1 is not reset. 1 I^{2}C module 1 is reset.
0	R0	RW	0	I^{2}C Module 0 Software Reset Value Description 0 I^{2}C module 0 is not reset. 1 I^{2}C module 0 is reset.

Register 48: Universal Serial Bus Software Reset (SRUSB), offset 0x528

The **SRUSB** register provides software the capability to reset the available USB module. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the USB module and has the same bit polarity as the corresponding **SRCRn** bits.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRUSB** register. While the **SRUSB** bit is 1, the peripheral is held in reset.

2. Software completes the reset process by clearing the **SRUSB** bit.

There may be latency from the clearing of the **SRUSB** bit to when the peripheral is ready for use. Software can check the corresponding **PRUSB** bit to be sure.

Important: This register should be used to reset the USB module. To support legacy software, the **SRCR2** register is available. Setting the `USB0` bit in the **SRCR2** register also resets the USB module. If the `USB0` bit is set by writing to the **SRCR2** register, it can be read back correctly when reading the **SRCR2** register. If software uses this register to reset the USB module, the write causes proper operation, but the value of the `USB0` bit is not reflected in the **SRCR2** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Universal Serial Bus Software Reset (SRUSB)

Base 0x400F.E000
Offset 0x528
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RW	0	USB Module Software Reset Value Description 0 USB module is not reset. 1 USB module is reset.

Register 49: Controller Area Network Software Reset (SRCAN), offset 0x534

The **SRCAN** register provides software the capability to reset the available CAN modules. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the CAN modules and has the same bit polarity as the corresponding **SRCRn** bits.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRCAN** register. While the **SRCAN** bit is 1, the peripheral is held in reset.

2. Software completes the reset process by clearing the **SRCAN** bit.

There may be latency from the clearing of the **SRCAN** bit to when the peripheral is ready for use. Software can check the corresponding **PRCAN** bit to be sure.

Important: This register should be used to reset the CAN modules. To support legacy software, the **SRCR0** register is available. Setting a bit in the **SRCR0** register also resets the corresponding module. Any bits that are changed by writing to the **SRCR0** register can be read back correctly when reading the SRCR0 register. If software uses this register to reset a legacy peripheral (such as CAN0), the write causes proper operation, but the value of that bit is not reflected in the **SRCR0** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Controller Area Network Software Reset (SRCAN)

Base 0x400F.E000
Offset 0x534
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RW	0	CAN Module 0 Software Reset

Value	Description
0	CAN module 0 is not reset.
1	CAN module 0 is reset.

Register 50: Analog-to-Digital Converter Software Reset (SRADC), offset 0x538

The **SRADC** register provides software the capability to reset the available ADC modules. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the ADC modules and has the same bit polarity as the corresponding **SRCRn** bits.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRADC** register. While the **SRADC** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRADC** bit.

There may be latency from the clearing of the **SRADC** bit to when the peripheral is ready for use. Software can check the corresponding **PRADC** bit to be sure.

Important: This register should be used to reset the ADC modules. To support legacy software, the **SRCR0** register is available. Setting a bit in the **SRCR0** register also resets the corresponding module. Any bits that are changed by writing to the **SRCR0** register can be read back correctly when reading the **SRCR0** register. If software uses this register to reset a legacy peripheral (such as ADC0), the write causes proper operation, but the value of that bit is not reflected in the **SRCR0** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Analog-to-Digital Converter Software Reset (SRADC)

Base 0x400F.E000
Offset 0x538
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	R1	RW	0	ADC Module 1 Software Reset

Value	Description
0	ADC module 1 is not reset.
1	ADC module 1 is reset.

Bit/Field	Name	Type	Reset	Description
0	R0	RW	0	ADC Module 0 Software Reset

Value	Description
0	ADC module 0 is not reset.
1	ADC module 0 is reset.

Register 51: Analog Comparator Software Reset (SRACMP), offset 0x53C

The **SRACMP** register provides software the capability to reset the available analog comparator module. This register provides the same capability as the legacy **Software Reset Control n SRCRn** registers specifically for the analog comparator module and has the same bit polarity as the corresponding **SRCRn** bits.

A block is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRACMP** register. While the **SRACMP** bit is 1, the module is held in reset.
2. Software completes the reset process by clearing the **SRACMP** bit.

There may be latency from the clearing of the **SRACMP** bit to when the module is ready for use. Software can check the corresponding **PRACMP** bit to be sure.

Important: This register should be used to reset the analog comparator module. To support legacy software, the **SRCR1** register is available. Setting any of the `COMPn` bits in the **SRCR0** register also resets the analog comparator module. If any of the `COMPn` bits are set by writing to the **SRCR1** register, it can be read back correctly when reading the **SRCR0** register. If software uses this register to reset the analog comparator module, the write causes proper operation, but the value of `R0` is not reflected by the `COMPn` bits in the **SRCR1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Analog Comparator Software Reset (SRACMP)

Base 0x400F.E000
Offset 0x53C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RW	0	Analog Comparator Module 0 Software Reset

Value	Description
0	Analog comparator module is not reset.
1	Analog comparator module is reset.

Register 52: EEPROM Software Reset (SREEPROM), offset 0x558

The **SREEPROM** register provides software the capability to reset the available EEPROM module.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SREEPROM** register. While the **SREEPROM** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SREEPROM** bit.

There may be latency from the clearing of the **SREEPROM** bit to when the peripheral is ready for use. Software can check the corresponding **PREEPROM** bit to be sure.

EEPROM Software Reset (SREEPROM)

Base 0x400F.E000
Offset 0x558
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RW	0	EEPROM Module Software Reset

Value	Description
0	EEPROM module is not reset.
1	EEPROM module is reset.

Register 53: 32/64-Bit Wide General-Purpose Timer Software Reset (SRWTIMER), offset 0x55C

The **SRWTIMER** register provides software the capability to reset the available 32/64-bit wide timer modules.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRWTIMER** register. While the **SRWTIMER** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRWTIMER** bit.

There may be latency from the clearing of the **SRWTIMER** bit to when the peripheral is ready for use. Software can check the corresponding **PRWTIMER** bit to be sure.

32/64-Bit Wide General-Purpose Timer Software Reset (SRWTIMER)

Base 0x400F.E000
Offset 0x55C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	R5	RW	0	32/64-Bit Wide General-Purpose Timer 5 Software Reset Value Description 0 32/64-bit wide general-purpose timer module 5 is not reset. 1 32/64-bit wide general-purpose timer module 5 is reset.
4	R4	RW	0	32/64-Bit Wide General-Purpose Timer 4 Software Reset Value Description 0 32/64-bit wide general-purpose timer module 4 is not reset. 1 32/64-bit wide general-purpose timer module 4 is reset.
3	R3	RW	0	32/64-Bit Wide General-Purpose Timer 3 Software Reset Value Description 0 32/64-bit wide general-purpose timer module 3 is not reset. 1 32/64-bit wide general-purpose timer module 3 is reset.

Bit/Field	Name	Type	Reset	Description
2	R2	RW	0	32/64-Bit Wide General-Purpose Timer 2 Software Reset Value Description 0 32/64-bit wide general-purpose timer module 2 is not reset. 1 32/64-bit wide general-purpose timer module 2 is reset.
1	R1	RW	0	32/64-Bit Wide General-Purpose Timer 1 Software Reset Value Description 0 32/64-bit wide general-purpose timer module 1 is not reset. 1 32/64-bit wide general-purpose timer module 1 is reset.
0	R0	RW	0	32/64-Bit Wide General-Purpose Timer 0 Software Reset Value Description 0 32/64-bit wide general-purpose timer module 0 is not reset. 1 32/64-bit wide general-purpose timer module 0 is reset.

Register 54: Watchdog Timer Run Mode Clock Gating Control (RCGCWD), offset 0x600

The **RCGCWD** register provides software the capability to enable and disable watchdog modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the watchdog modules. To support legacy software, the **RCGC0** register is available. A write to the **RCGC0** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **RCGC0** register can be read back correctly with a read of the **RCGC0** register. If software uses this register to write a legacy peripheral (such as Watchdog 0), the write causes proper operation, but the value of that bit is not reflected in the **RCGC0** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Watchdog Timer Run Mode Clock Gating Control (RCGCWD)

Base 0x400F.E000
Offset 0x600
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	R1	RW	0	Watchdog Timer 1 Run Mode Clock Gating Control Value Description 0 Watchdog module 1 is disabled. 1 Enable and provide a clock to Watchdog module 1 in Run mode.
0	R0	RW	0	Watchdog Timer 0 Run Mode Clock Gating Control Value Description 0 Watchdog module 0 is disabled. 1 Enable and provide a clock to Watchdog module 0 in Run mode.

Register 55: 16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER), offset 0x604

The **RCGCTIMER** register provides software the capability to enable and disable 16/32-bit timer modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the timer modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the timer modules. To support legacy software, the **RCGC1** register is available. A write to the **RCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **RCGC1** register can be read back correctly with a read of the **RCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as Timer 0), the write causes proper operation, but the value of that bit is not reflected in the **RCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER)

Base 0x400F.E000
Offset 0x604
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	R5	RW	0	16/32-Bit General-Purpose Timer 5 Run Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 5 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 5 in Run mode.

Bit/Field	Name	Type	Reset	Description
4	R4	RW	0	16/32-Bit General-Purpose Timer 4 Run Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 4 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 4 in Run mode.
3	R3	RW	0	16/32-Bit General-Purpose Timer 3 Run Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 3 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 3 in Run mode.
2	R2	RW	0	16/32-Bit General-Purpose Timer 2 Run Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 2 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 2 in Run mode.
1	R1	RW	0	16/32-Bit General-Purpose Timer 1 Run Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 1 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 1 in Run mode.
0	R0	RW	0	16/32-Bit General-Purpose Timer 0 Run Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 0 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 0 in Run mode.

Register 56: General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO), offset 0x608

The **RCGCGPIO** register provides software the capability to enable and disable GPIO modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the GPIO modules. To support legacy software, the **RCGC2** register is available. A write to the **RCGC2** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **RCGC2** register can be read back correctly with a read of the **RCGC2** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as GPIO A), the write causes proper operation, but the value of that bit is not reflected in the **RCGC2** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO)

Base 0x400F.E000
Offset 0x608
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									R6	R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	R6	RW	0	GPIO Port G Run Mode Clock Gating Control Value Description 0 GPIO Port G is disabled. 1 Enable and provide a clock to GPIO Port G in Run mode.
5	R5	RW	0	GPIO Port F Run Mode Clock Gating Control Value Description 0 GPIO Port F is disabled. 1 Enable and provide a clock to GPIO Port F in Run mode.

Bit/Field	Name	Type	Reset	Description
4	R4	RW	0	GPIO Port E Run Mode Clock Gating Control Value Description 0 GPIO Port E is disabled. 1 Enable and provide a clock to GPIO Port E in Run mode.
3	R3	RW	0	GPIO Port D Run Mode Clock Gating Control Value Description 0 GPIO Port D is disabled. 1 Enable and provide a clock to GPIO Port D in Run mode.
2	R2	RW	0	GPIO Port C Run Mode Clock Gating Control Value Description 0 GPIO Port C is disabled. 1 Enable and provide a clock to GPIO Port C in Run mode.
1	R1	RW	0	GPIO Port B Run Mode Clock Gating Control Value Description 0 GPIO Port B is disabled. 1 Enable and provide a clock to GPIO Port B in Run mode.
0	R0	RW	0	GPIO Port A Run Mode Clock Gating Control Value Description 0 GPIO Port A is disabled. 1 Enable and provide a clock to GPIO Port A in Run mode.

Register 57: Micro Direct Memory Access Run Mode Clock Gating Control (RCGCDMA), offset 0x60C

The **RCGCDMA** register provides software the capability to enable and disable the µDMA module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the µDMA module. To support legacy software, the **RCGC2** register is available. A write to the `UDMA` bit in the **RCGC2** register also writes the `R0` bit in this register. If the `UDMA` bit is changed by writing to the **RCGC2** register, it can be read back correctly with a read of the **RCGC2** register. If software uses this register to control the clock for the µDMA module, the write causes proper operation, but the `UDMA` bit in the **RCGC2** register does not reflect the value of the `R0` bit. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Micro Direct Memory Access Run Mode Clock Gating Control (RCGCDMA)

Base 0x400F.E000
Offset 0x60C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RW	0	µDMA Module Run Mode Clock Gating Control

Value	Description
0	µDMA module is disabled.
1	Enable and provide a clock to the µDMA module in Run mode.

Register 58: Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control (RCGCUART), offset 0x618

The **RCGCUART** register provides software the capability to enable and disable the UART modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the UART modules. To support legacy software, the **RCGC1** register is available. A write to the **RCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **RCGC1** register can be read back correctly with a read of the **RCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as UART0), the write causes proper operation, but the value of that bit is not reflected in the **RCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control (RCGCUART)

Base 0x400F.E000
Offset 0x618
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								R7	R6	R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	R7	RW	0	UART Module 7 Run Mode Clock Gating Control Value Description 0 UART module 7 is disabled. 1 Enable and provide a clock to UART module 7 in Run mode.
6	R6	RW	0	UART Module 6 Run Mode Clock Gating Control Value Description 0 UART module 6 is disabled. 1 Enable and provide a clock to UART module 6 in Run mode.

Bit/Field	Name	Type	Reset	Description
5	R5	RW	0	UART Module 5 Run Mode Clock Gating Control Value Description 0 UART module 5 is disabled. 1 Enable and provide a clock to UART module 5 in Run mode.
4	R4	RW	0	UART Module 4 Run Mode Clock Gating Control Value Description 0 UART module 4 is disabled. 1 Enable and provide a clock to UART module 4 in Run mode.
3	R3	RW	0	UART Module 3 Run Mode Clock Gating Control Value Description 0 UART module 3 is disabled. 1 Enable and provide a clock to UART module 3 in Run mode.
2	R2	RW	0	UART Module 2 Run Mode Clock Gating Control Value Description 0 UART module 2 is disabled. 1 Enable and provide a clock to UART module 2 in Run mode.
1	R1	RW	0	UART Module 1 Run Mode Clock Gating Control Value Description 0 UART module 1 is disabled. 1 Enable and provide a clock to UART module 1 in Run mode.
0	R0	RW	0	UART Module 0 Run Mode Clock Gating Control Value Description 0 UART module 0 is disabled. 1 Enable and provide a clock to UART module 0 in Run mode.

Register 59: Synchronous Serial Interface Run Mode Clock Gating Control (RCGCSSI), offset 0x61C

The **RCGCSSI** register provides software the capability to enable and disable the SSI modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the SSI modules. To support legacy software, the **RCGC1** register is available. A write to the **RCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **RCGC1** register can be read back correctly with a read of the **RCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as SSI0), the write causes proper operation, but the value of that bit is not reflected in the **RCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Synchronous Serial Interface Run Mode Clock Gating Control (RCGCSSI)

Base 0x400F.E000
Offset 0x61C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	R3	RW	0	SSI Module 3 Run Mode Clock Gating Control Value Description 0 SSI module 3 is disabled. 1 Enable and provide a clock to SSI module 3 in Run mode.
2	R2	RW	0	SSI Module 2 Run Mode Clock Gating Control Value Description 0 SSI module 2 is disabled. 1 Enable and provide a clock to SSI module 2 in Run mode.

Bit/Field	Name	Type	Reset	Description
1	R1	RW	0	SSI Module 1 Run Mode Clock Gating Control Value Description 0 SSI module 1 is disabled. 1 Enable and provide a clock to SSI module 1 in Run mode.
0	R0	RW	0	SSI Module 0 Run Mode Clock Gating Control Value Description 0 SSI module 0 is disabled. 1 Enable and provide a clock to SSI module 0 in Run mode.

Register 60: Inter-Integrated Circuit Run Mode Clock Gating Control (RCGCI2C), offset 0x620

The **RCGCI2C** register provides software the capability to enable and disable the I^{2}C modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the I^{2}C modules. To support legacy software, the **RCGC1** register is available. A write to the **RCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **RCGC1** register can be read back correctly with a read of the **RCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as I2C0), the write causes proper operation, but the value of that bit is not reflected in the **RCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Inter-Integrated Circuit Run Mode Clock Gating Control (RCGCI2C)

Base 0x400F.E000
Offset 0x620
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	R5	RW	0	I^{2}C Module 5 Run Mode Clock Gating Control Value Description 0 I^{2}C module 5 is disabled. 1 Enable and provide a clock to I^{2}C module 5 in Run mode.
4	R4	RW	0	I^{2}C Module 4 Run Mode Clock Gating Control Value Description 0 I^{2}C module 4 is disabled. 1 Enable and provide a clock to I^{2}C module 4 in Run mode.

Bit/Field	Name	Type	Reset	Description
3	R3	RW	0	I^{2}C Module 3 Run Mode Clock Gating Control Value Description 0 I^{2}C module 3 is disabled. 1 Enable and provide a clock to I^{2}C module 3 in Run mode.
2	R2	RW	0	I^{2}C Module 2 Run Mode Clock Gating Control Value Description 0 I^{2}C module 2 is disabled. 1 Enable and provide a clock to I^{2}C module 2 in Run mode.
1	R1	RW	0	I^{2}C Module 1 Run Mode Clock Gating Control Value Description 0 I^{2}C module 1 is disabled. 1 Enable and provide a clock to I^{2}C module 1 in Run mode.
0	R0	RW	0	I^{2}C Module 0 Run Mode Clock Gating Control Value Description 0 I^{2}C module 0 is disabled. 1 Enable and provide a clock to I^{2}C module 0 in Run mode.

Register 61: Universal Serial Bus Run Mode Clock Gating Control (RCGCUSB), offset 0x628

The **RCGCUSB** register provides software the capability to enable and disable the USB module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the USB module. To support legacy software, the **RCGC2** register is available. A write to the `USB0` bit in the **RCGC2** register also writes the `R0` bit in this register. If the `USB0` bit is changed by writing to the **RCGC2** register, it can be read back correctly with a read of the **RCGC2** register. If software uses this register to control the clock for the USB module, the write causes proper operation, but the `USB0` bit in the **RCGC2** register does not reflect the value of the `R0` bit. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Universal Serial Bus Run Mode Clock Gating Control (RCGCUSB)

Base 0x400F.E000
Offset 0x628
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RW	0	USB Module Run Mode Clock Gating Control Value Description 0 USB module is disabled. 1 Enable and provide a clock to the USB module in Run mode.

Register 62: Controller Area Network Run Mode Clock Gating Control (RCGCCAN), offset 0x634

The **RCGCCAN** register provides software the capability to enable and disable the CAN modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the CAN modules. To support legacy software, the RCGC0 register is available. A write to the RCGC0 register also writes the corresponding bit in this register. Any bits that are changed by writing to the RCGC0 register can be read back correctly with a read of the RCGC0 register. If software uses this register to write a legacy peripheral (such as CAN0), the write causes proper operation, but the value of that bit is not reflected in the RCGC0 register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Controller Area Network Run Mode Clock Gating Control (RCGCCAN)

Base 0x400F.E000
Offset 0x634
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RW	0	CAN Module 0 Run Mode Clock Gating Control

Value	Description
0	CAN module 0 is disabled.
1	Enable and provide a clock to CAN module 0 in Run mode.

Register 63: Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC), offset 0x638

The **RCGCADC** register provides software the capability to enable and disable the ADC modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the ADC modules. To support legacy software, the RCGC0 register is available. A write to the RCGC0 register also writes the corresponding bit in this register. Any bits that are changed by writing to the RCGC0 register can be read back correctly with a read of the RCGC0 register. If software uses this register to write a legacy peripheral (such as ADC0), the write causes proper operation, but the value of that bit is not reflected in the RCGC0 register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC)

Base 0x400F.E000
Offset 0x638
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	R1	RW	0	ADC Module 1 Run Mode Clock Gating Control Value Description 0 ADC module 1 is disabled. 1 Enable and provide a clock to ADC module 1 in Run mode.
0	R0	RW	0	ADC Module 0 Run Mode Clock Gating Control Value Description 0 ADC module 0 is disabled. 1 Enable and provide a clock to ADC module 0 in Run mode.

Register 64: Analog Comparator Run Mode Clock Gating Control (RCGCACMP), offset 0x63C

The **RCGCACMP** register provides software the capability to enable and disable the analog comparator module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **RCGCn** bits.

Important: This register should be used to control the clocking for the analog comparator module. To support legacy software, the RCGC1 register is available. Setting any of the `COMPn` bits in the RCGC1 register also sets the `R0` bit in this register. If any of the `COMPn` bits are set by writing to the RCGC1 register, it can be read back correctly when reading the RCGC1 register. If software uses this register to change the clocking for the analog comparator module, the write causes proper operation, but the value `R0` is not reflected by the `COMPn` bits in the RCGC1 register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Analog Comparator Run Mode Clock Gating Control (RCGCACMP)

Base 0x400F.E000
Offset 0x63C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RW	0	Analog Comparator Module 0 Run Mode Clock Gating Control

Value	Description
0	Analog comparator module is disabled.
1	Enable and provide a clock to the analog comparator module in Run mode.

Register 65: EEPROM Run Mode Clock Gating Control (RCGCEEPROM), offset 0x658

The **RCGCEEPROM** register provides software the capability to enable and disable the EEPROM module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

EEPROM Run Mode Clock Gating Control (RCGCEEPROM)

Base 0x400F.E000
Offset 0x658
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RW	0	EEPROM Module Run Mode Clock Gating Control Value Description 0 EEPROM module is disabled. 1 Enable and provide a clock to the EEPROM module in Run mode.

Register 66: 32/64-Bit Wide General-Purpose Timer Run Mode Clock Gating Control (RCGCWTIMER), offset 0x65C

The **RCGCWTIMER** register provides software the capability to enable and disable 3264-bit timer modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault. This register provides the same capability as the legacy **Run Mode Clock Gating Control Register n RCGCn** registers specifically for the timer modules and has the same bit polarity as the corresponding **RCGCn** bits.

32/64-Bit Wide General-Purpose Timer Run Mode Clock Gating Control (RCGCWTIMER)

Base 0x400F.E000
Offset 0x65C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	R5	RW	0	32/64-Bit Wide General-Purpose Timer 5 Run Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 5 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 5 in Run mode.
4	R4	RW	0	32/64-Bit Wide General-Purpose Timer 4 Run Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 4 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 4 in Run mode.
3	R3	RW	0	32/64-Bit Wide General-Purpose Timer 3 Run Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 3 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 3 in Run mode.

Bit/Field	Name	Type	Reset	Description
2	R2	RW	0	32/64-Bit Wide General-Purpose Timer 2 Run Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 2 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 2 in Run mode.
1	R1	RW	0	32/64-Bit Wide General-Purpose Timer 1 Run Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 1 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 1 in Run mode.
0	R0	RW	0	32/64-Bit Wide General-Purpose Timer 0 Run Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 0 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 0 in Run mode.

Register 67: Watchdog Timer Sleep Mode Clock Gating Control (SCGCWD), offset 0x700

The **SCGCWD** register provides software the capability to enable and disable watchdog modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the watchdog modules. To support legacy software, the **SCGC0** register is available. A write to the **SCGC0** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **SCGC0** register can be read back correctly with a read of the **SCGC0** register. If software uses this register to write a legacy peripheral (such as Watchdog 0), the write causes proper operation, but the value of that bit is not reflected in the **SCGC0** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Watchdog Timer Sleep Mode Clock Gating Control (SCGCWD)

Base 0x400F.E000
Offset 0x700
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														S1	S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	S1	RW	0	Watchdog Timer 1 Sleep Mode Clock Gating Control Value Description 0 Watchdog module 1 is disabled. 1 Enable and provide a clock to Watchdog module 1 in sleep mode.
0	S0	RW	0	Watchdog Timer 0 Sleep Mode Clock Gating Control Value Description 0 Watchdog module 0 is disabled. 1 Enable and provide a clock to Watchdog module 0 in sleep mode.

Register 68: 16/32-Bit General-Purpose Timer Sleep Mode Clock Gating Control (SCGCTIMER), offset 0x704

The **SCGCTIMER** register provides software the capability to enable and disable 16/32-bit timer modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the timer modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the timer modules. To support legacy software, the **SCGC1** register is available. A write to the **SCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **SCGC1** register can be read back correctly with a read of the **SCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as Timer 0), the write causes proper operation, but the value of that bit is not reflected in the **SCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

16/32-Bit General-Purpose Timer Sleep Mode Clock Gating Control (SCGCTIMER)

Base 0x400F.E000
Offset 0x704
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										S5	S4	S3	S2	S1	S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	S5	RW	0	16/32-Bit General-Purpose Timer 5 Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 5 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 5 in sleep mode.
4	S4	RW	0	16/32-Bit General-Purpose Timer 4 Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 4 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 4 in sleep mode.

Bit/Field	Name	Type	Reset	Description
3	S3	RW	0	16/32-Bit General-Purpose Timer 3 Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 3 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 3 in sleep mode.
2	S2	RW	0	16/32-Bit General-Purpose Timer 2 Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 2 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 2 in sleep mode.
1	S1	RW	0	16/32-Bit General-Purpose Timer 1 Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 1 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 1 in sleep mode.
0	S0	RW	0	16/32-Bit General-Purpose Timer 0 Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 0 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 0 in sleep mode.

Register 69: General-Purpose Input/Output Sleep Mode Clock Gating Control (SCGCGPIO), offset 0x708

The **SCGCGPIO** register provides software the capability to enable and disable GPIO modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the GPIO modules. To support legacy software, the **SCGC2** register is available. A write to the **SCGC2** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **SCGC2** register can be read back correctly with a read of the **SCGC2** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as GPIO A), the write causes proper operation, but the value of that bit is not reflected in the **SCGC2** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

General-Purpose Input/Output Sleep Mode Clock Gating Control (SCGCGPIO)

Base 0x400F.E000
Offset 0x708
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									S6	S5	S4	S3	S2	S1	S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	S6	RW	0	GPIO Port G Sleep Mode Clock Gating Control Value Description 0 GPIO Port G is disabled. 1 Enable and provide a clock to GPIO Port G in sleep mode.
5	S5	RW	0	GPIO Port F Sleep Mode Clock Gating Control Value Description 0 GPIO Port F is disabled. 1 Enable and provide a clock to GPIO Port F in sleep mode.

Bit/Field	Name	Type	Reset	Description
4	S4	RW	0	GPIO Port E Sleep Mode Clock Gating Control Value Description 0 GPIO Port E is disabled. 1 Enable and provide a clock to GPIO Port E in sleep mode.
3	S3	RW	0	GPIO Port D Sleep Mode Clock Gating Control Value Description 0 GPIO Port D is disabled. 1 Enable and provide a clock to GPIO Port D in sleep mode.
2	S2	RW	0	GPIO Port C Sleep Mode Clock Gating Control Value Description 0 GPIO Port C is disabled. 1 Enable and provide a clock to GPIO Port C in sleep mode.
1	S1	RW	0	GPIO Port B Sleep Mode Clock Gating Control Value Description 0 GPIO Port B is disabled. 1 Enable and provide a clock to GPIO Port B in sleep mode.
0	S0	RW	0	GPIO Port A Sleep Mode Clock Gating Control Value Description 0 GPIO Port A is disabled. 1 Enable and provide a clock to GPIO Port A in sleep mode.

Register 70: Micro Direct Memory Access Sleep Mode Clock Gating Control (SCGCDMA), offset 0x70C

The **SCGCDMA** register provides software the capability to enable and disable the μDMA module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the μDMA module. To support legacy software, the **SCGC2** register is available. A write to the `UDMA` bit in the **SCGC2** register also writes the `S0` bit in this register. If the `UDMA` bit is changed by writing to the **SCGC2** register, it can be read back correctly with a read of the **SCGC2** register. If software uses this register to control the clock for the μDMA module, the write causes proper operation, but the `UDMA` bit in the **SCGC2** register does not reflect the value of the `S0` bit. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Micro Direct Memory Access Sleep Mode Clock Gating Control (SCGCDMA)

Base 0x400F.E000
Offset 0x70C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0	RW	0	μDMA Module Sleep Mode Clock Gating Control

Value	Description
0	μDMA module is disabled.
1	Enable and provide a clock to the μDMA module in sleep mode.

Register 71: Universal Asynchronous Receiver/Transmitter Sleep Mode Clock Gating Control (SCGCUART), offset 0x718

The **SCGCUART** register provides software the capability to enable and disable the UART modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the UART modules. To support legacy software, the **SCGC1** register is available. A write to the **SCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **SCGC1** register can be read back correctly with a read of the **SCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as UART0), the write causes proper operation, but the value of that bit is not reflected in the **SCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Universal Asynchronous Receiver/Transmitter Sleep Mode Clock Gating Control (SCGCUART)

Base 0x400F.E000
Offset 0x718
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								S7	S6	S5	S4	S3	S2	S1	S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	S7	RW	0	UART Module 7 Sleep Mode Clock Gating Control Value Description 0 UART module 7 is disabled. 1 Enable and provide a clock to UART module 7 in sleep mode.
6	S6	RW	0	UART Module 6 Sleep Mode Clock Gating Control Value Description 0 UART module 6 is disabled. 1 Enable and provide a clock to UART module 6 in sleep mode.

Bit/Field	Name	Type	Reset	Description
5	S5	RW	0	UART Module 5 Sleep Mode Clock Gating Control Value Description 0 UART module 5 is disabled. 1 Enable and provide a clock to UART module 5 in sleep mode.
4	S4	RW	0	UART Module 4 Sleep Mode Clock Gating Control Value Description 0 UART module 4 is disabled. 1 Enable and provide a clock to UART module 4 in sleep mode.
3	S3	RW	0	UART Module 3 Sleep Mode Clock Gating Control Value Description 0 UART module 3 is disabled. 1 Enable and provide a clock to UART module 3 in sleep mode.
2	S2	RW	0	UART Module 2 Sleep Mode Clock Gating Control Value Description 0 UART module 2 is disabled. 1 Enable and provide a clock to UART module 2 in sleep mode.
1	S1	RW	0	UART Module 1 Sleep Mode Clock Gating Control Value Description 0 UART module 1 is disabled. 1 Enable and provide a clock to UART module 1 in sleep mode.
0	S0	RW	0	UART Module 0 Sleep Mode Clock Gating Control Value Description 0 UART module 0 is disabled. 1 Enable and provide a clock to UART module 0 in sleep mode.

Register 72: Synchronous Serial Interface Sleep Mode Clock Gating Control (SCGCSSI), offset 0x71C

The **SCGCSSI** register provides software the capability to enable and disable the SSI modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the SSI modules. To support legacy software, the **SCGC1** register is available. A write to the **SCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **SCGC1** register can be read back correctly with a read of the **SCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as SSI0), the write causes proper operation, but the value of that bit is not reflected in the **SCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Synchronous Serial Interface Sleep Mode Clock Gating Control (SCGCSSI)

Base 0x400F.E000
Offset 0x71C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												S3	S2	S1	S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	S3	RW	0	SSI Module 3 Sleep Mode Clock Gating Control Value Description 0 SSI module 3 is disabled. 1 Enable and provide a clock to SSI module 3 in sleep mode.
2	S2	RW	0	SSI Module 2 Sleep Mode Clock Gating Control Value Description 0 SSI module 2 is disabled. 1 Enable and provide a clock to SSI module 2 in sleep mode.

Bit/Field	Name	Type	Reset	Description
1	S1	RW	0	SSI Module 1 Sleep Mode Clock Gating Control Value Description 0 SSI module 1 is disabled. 1 Enable and provide a clock to SSI module 1 in sleep mode.
0	S0	RW	0	SSI Module 0 Sleep Mode Clock Gating Control Value Description 0 SSI module 0 is disabled. 1 Enable and provide a clock to SSI module 0 in sleep mode.

Register 73: Inter-Integrated Circuit Sleep Mode Clock Gating Control (SCGCI2C), offset 0x720

The **SCGCI2C** register provides software the capability to enable and disable the I^{2}C modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the I^{2}C modules. To support legacy software, the **SCGC1** register is available. A write to the **SCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **SCGC1** register can be read back correctly with a read of the **SCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as I^{2}C0), the write causes proper operation, but the value of that bit is not reflected in the **SCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Inter-Integrated Circuit Sleep Mode Clock Gating Control (SCGCI2C)

Base 0x400F.E000
Offset 0x720
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										S5	S4	S3	S2	S1	S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	S5	RW	0	I^{2}C Module 5 Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 5 is disabled. 1 Enable and provide a clock to I^{2}C module 5 in sleep mode.
4	S4	RW	0	I^{2}C Module 4 Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 4 is disabled. 1 Enable and provide a clock to I^{2}C module 4 in sleep mode.

Bit/Field	Name	Type	Reset	Description
3	S3	RW	0	I^{2}C Module 3 Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 3 is disabled. 1 Enable and provide a clock to I^{2}C module 3 in sleep mode.
2	S2	RW	0	I^{2}C Module 2 Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 2 is disabled. 1 Enable and provide a clock to I^{2}C module 2 in sleep mode.
1	S1	RW	0	I^{2}C Module 1 Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 1 is disabled. 1 Enable and provide a clock to I^{2}C module 1 in sleep mode.
0	S0	RW	0	I^{2}C Module 0 Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 0 is disabled. 1 Enable and provide a clock to I^{2}C module 0 in sleep mode.

Register 74: Universal Serial Bus Sleep Mode Clock Gating Control (SCGCUSB), offset 0x728

The **SCGCUSB** register provides software the capability to enable and disable the USB module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the USB module. To support legacy software, the **SCGC2** register is available. A write to the `USB0` bit in the **SCGC2** register also writes the `S0` bit in this register. If the `USB0` bit is changed by writing to the **SCGC2** register, it can be read back correctly with a read of the **SCGC2** register. If software uses this register to control the clock for the USB module, the write causes proper operation, but the `USB0` bit in the **SCGC2** register does not reflect the value of the `S0` bit. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Universal Serial Bus Sleep Mode Clock Gating Control (SCGCUSB)

Base 0x400F.E000
Offset 0x728
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0	RW	0	USB Module Sleep Mode Clock Gating Control

Value	Description
0	USB module is disabled.
1	Enable and provide a clock to the USB module in sleep mode.

Register 75: Controller Area Network Sleep Mode Clock Gating Control (SCGCCAN), offset 0x734

The **SCGCCAN** register provides software the capability to enable and disable the CAN modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the CAN modules. To support legacy software, the **SCGC0** register is available. A write to the **SCGC0** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **SCGC0** register can be read back correctly with a read of the **SCGC0** register. If software uses this register to write a legacy peripheral (such as CAN0), the write causes proper operation, but the value of that bit is not reflected in the **SCGC0** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Controller Area Network Sleep Mode Clock Gating Control (SCGCCAN)

Base 0x400F.E000
Offset 0x734
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0	RW	0	CAN Module 0 Sleep Mode Clock Gating Control

Value	Description
0	CAN module 0 is disabled.
1	Enable and provide a clock to CAN module 0 in sleep mode.

Register 76: Analog-to-Digital Converter Sleep Mode Clock Gating Control (SCGCADC), offset 0x738

The **SCGCADC** register provides software the capability to enable and disable the ADC modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the ADC modules. To support legacy software, the **SCGC0** register is available. A write to the **SCGC0** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **SCGC0** register can be read back correctly with a read of the **SCGC0** register. If software uses this register to write a legacy peripheral (such as ADC0), the write causes proper operation, but the value of that bit is not reflected in the **SCGC0** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Analog-to-Digital Converter Sleep Mode Clock Gating Control (SCGCADC)

Base 0x400F.E000
Offset 0x738
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														S1	S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	S1	RW	0	ADC Module 1 Sleep Mode Clock Gating Control Value Description 0 ADC module 1 is disabled. 1 Enable and provide a clock to ADC module 1 in sleep mode.
0	S0	RW	0	ADC Module 0 Sleep Mode Clock Gating Control Value Description 0 ADC module 0 is disabled. 1 Enable and provide a clock to ADC module 0 in sleep mode.

Register 77: Analog Comparator Sleep Mode Clock Gating Control (SCGCACMP), offset 0x73C

The **SCGCACMP** register provides software the capability to enable and disable the analog comparator module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **SCGCn** bits.

Important: This register should be used to control the clocking for the analog comparator module. To support legacy software, the **SCGC1** register is available. Setting any of the `COMPn` bits in the SCGC1 register also sets the `S0` bit in this register. If any of the `COMPn` bits are set by writing to the **SCGC1** register, it can be read back correctly when reading the **SCGC1** register. If software uses this register to change the clocking for the analog comparator module, the write causes proper operation, but the value `S0` is not reflected by the `COMPn` bits in the **SCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Analog Comparator Sleep Mode Clock Gating Control (SCGCACMP)

Base 0x400F.E000
Offset 0x73C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0	RW	0	Analog Comparator Module 0 Sleep Mode Clock Gating Control

Value	Description
0	Analog comparator module is disabled.
1	Enable and provide a clock to the analog comparator module in sleep mode.

Register 78: EEPROM Sleep Mode Clock Gating Control (SCGCEEPROM), offset 0x758

The **SCGCEEPROM** register provides software the capability to enable and disable the EEPROM module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

EEPROM Sleep Mode Clock Gating Control (SCGCEEPROM)

Base 0x400F.E000
Offset 0x758
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0	RW	0	EEPROM Module Sleep Mode Clock Gating Control Value Description 0 EEPROM module is disabled. 1 Enable and provide a clock to the EEPROM module in sleep mode.

Register 79: 32/64-Bit Wide General-Purpose Timer Sleep Mode Clock Gating Control (SCGCWTIMER), offset 0x75C

The **SCGCWTIMER** register provides software the capability to enable and disable 3264-bit timer modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Sleep Mode Clock Gating Control Register n SCGCn** registers specifically for the timer modules and has the same bit polarity as the corresponding **SCGCn** bits.

32/64-Bit Wide General-Purpose Timer Sleep Mode Clock Gating Control (SCGCWTIMER)

Base 0x400F.E000
Offset 0x75C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										S5	S4	S3	S2	S1	S0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	S5	RW	0	32/64-Bit Wide General-Purpose Timer 5 Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 5 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 5 in sleep mode.
4	S4	RW	0	32/64-Bit Wide General-Purpose Timer 4 Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 4 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 4 in sleep mode.
3	S3	RW	0	32/64-Bit Wide General-Purpose Timer 3 Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 3 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 3 in sleep mode.

Bit/Field	Name	Type	Reset	Description
2	S2	RW	0	32/64-Bit Wide General-Purpose Timer 2 Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 2 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 2 in sleep mode.
1	S1	RW	0	32/64-Bit Wide General-Purpose Timer 1 Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 1 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 1 in sleep mode.
0	S0	RW	0	32/64-Bit Wide General-Purpose Timer 0 Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 0 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 0 in sleep mode.

Register 80: Watchdog Timer Deep-Sleep Mode Clock Gating Control (DCGCWD), offset 0x800

The **DCGCWD** register provides software the capability to enable and disable watchdog modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the watchdog modules. To support legacy software, the **DCGC0** register is available. A write to the **DCGC0** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **DCGC0** register can be read back correctly with a read of the **DCGC0** register. If software uses this register to write a legacy peripheral (such as Watchdog 0), the write causes proper operation, but the value of that bit is not reflected in the **DCGC0** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Watchdog Timer Deep-Sleep Mode Clock Gating Control (DCGCWD)

Base 0x400F.E000
Offset 0x800
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														D1	D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	D1	RW	0	Watchdog Timer 1 Deep-Sleep Mode Clock Gating Control Value Description 0 Watchdog module 1 is disabled. 1 Enable and provide a clock to Watchdog module 1 in deep-sleep mode.
0	D0	RW	0	Watchdog Timer 0 Deep-Sleep Mode Clock Gating Control Value Description 0 Watchdog module 0 is disabled. 1 Enable and provide a clock to Watchdog module 0 in deep-sleep mode.

Register 81: 16/32-Bit General-Purpose Timer Deep-Sleep Mode Clock Gating Control (DCGCTIMER), offset 0x804

The **DCGCTIMER** register provides software the capability to enable and disable 16/32-bit timer modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the timer modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the timer modules. To support legacy software, the **DCGC1** register is available. A write to the **DCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **DCGC1** register can be read back correctly with a read of the **DCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as Timer 0), the write causes proper operation, but the value of that bit is not reflected in the **DCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

16/32-Bit General-Purpose Timer Deep-Sleep Mode Clock Gating Control (DCGCTIMER)

Base 0x400F.E000
Offset 0x804
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										D5	D4	D3	D2	D1	D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	D5	RW	0	16/32-Bit General-Purpose Timer 5 Deep-Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 5 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 5 in deep-sleep mode.

Bit/Field	Name	Type	Reset	Description
4	D4	RW	0	16/32-Bit General-Purpose Timer 4 Deep-Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 4 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 4 in deep-sleep mode.
3	D3	RW	0	16/32-Bit General-Purpose Timer 3 Deep-Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 3 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 3 in deep-sleep mode.
2	D2	RW	0	16/32-Bit General-Purpose Timer 2 Deep-Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 2 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 2 in deep-sleep mode.
1	D1	RW	0	16/32-Bit General-Purpose Timer 1 Deep-Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 1 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 1 in deep-sleep mode.
0	D0	RW	0	16/32-Bit General-Purpose Timer 0 Deep-Sleep Mode Clock Gating Control Value Description 0 16/32-bit general-purpose timer module 0 is disabled. 1 Enable and provide a clock to 16/32-bit general-purpose timer module 0 in deep-sleep mode.

Register 82: General-Purpose Input/Output Deep-Sleep Mode Clock Gating Control (DCGCGPIO), offset 0x808

The **DCGCGPIO** register provides software the capability to enable and disable GPIO modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the GPIO modules. To support legacy software, the **DCGC2** register is available. A write to the **DCGC2** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **DCGC2** register can be read back correctly with a read of the **DCGC2** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as GPIO A), the write causes proper operation, but the value of that bit is not reflected in the **DCGC2** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

General-Purpose Input/Output Deep-Sleep Mode Clock Gating Control (DCGCGPIO)

Base 0x400F.E000
Offset 0x808
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									D6	D5	D4	D3	D2	D1	D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	D6	RW	0	GPIO Port G Deep-Sleep Mode Clock Gating Control Value Description 0 GPIO Port G is disabled. 1 Enable and provide a clock to GPIO Port G in deep-sleep mode.
5	D5	RW	0	GPIO Port F Deep-Sleep Mode Clock Gating Control Value Description 0 GPIO Port F is disabled. 1 Enable and provide a clock to GPIO Port F in deep-sleep mode.

Bit/Field	Name	Type	Reset	Description
4	D4	RW	0	GPIO Port E Deep-Sleep Mode Clock Gating Control Value Description 0 GPIO Port E is disabled. 1 Enable and provide a clock to GPIO Port E in deep-sleep mode.
3	D3	RW	0	GPIO Port D Deep-Sleep Mode Clock Gating Control Value Description 0 GPIO Port D is disabled. 1 Enable and provide a clock to GPIO Port D in deep-sleep mode.
2	D2	RW	0	GPIO Port C Deep-Sleep Mode Clock Gating Control Value Description 0 GPIO Port C is disabled. 1 Enable and provide a clock to GPIO Port C in deep-sleep mode.
1	D1	RW	0	GPIO Port B Deep-Sleep Mode Clock Gating Control Value Description 0 GPIO Port B is disabled. 1 Enable and provide a clock to GPIO Port B in deep-sleep mode.
0	D0	RW	0	GPIO Port A Deep-Sleep Mode Clock Gating Control Value Description 0 GPIO Port A is disabled. 1 Enable and provide a clock to GPIO Port A in deep-sleep mode.

Register 83: Micro Direct Memory Access Deep-Sleep Mode Clock Gating Control (DCGCDMA), offset 0x80C

The **DCGCDMA** register provides software the capability to enable and disable the µDMA module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the µDMA module. To support legacy software, the **DCGC2** register is available. A write to the `UDMA` bit in the **DCGC2** register also writes the `D0` bit in this register. If the `UDMA` bit is changed by writing to the **DCGC2** register, it can be read back correctly with a read of the **DCGC2** register. If software uses this register to control the clock for the µDMA module, the write causes proper operation, but the `UDMA` bit in the **DCGC2** register does not reflect the value of the `D0` bit. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Micro Direct Memory Access Deep-Sleep Mode Clock Gating Control (DCGCDMA)

Base 0x400F.E000
Offset 0x80C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	D0	RW	0	µDMA Module Deep-Sleep Mode Clock Gating Control Value Description 0 µDMA module is disabled. 1 Enable and provide a clock to the µDMA module in deep-sleep mode.

Register 84: Universal Asynchronous Receiver/Transmitter Deep-Sleep Mode Clock Gating Control (DCGCUART), offset 0x818

The **DCGCUART** register provides software the capability to enable and disable the UART modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the UART modules. To support legacy software, the **DCGC1** register is available. A write to the **DCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **DCGC1** register can be read back correctly with a read of the **DCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as UART0), the write causes proper operation, but the value of that bit is not reflected in the **DCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Universal Asynchronous Receiver/Transmitter Deep-Sleep Mode Clock Gating Control (DCGCUART)

Base 0x400F.E000
Offset 0x818
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								D7	D6	D5	D4	D3	D2	D1	D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	D7	RW	0	UART Module 7 Deep-Sleep Mode Clock Gating Control Value Description 0 UART module 7 is disabled. 1 Enable and provide a clock to UART module 7 in deep-sleep mode.
6	D6	RW	0	UART Module 6 Deep-Sleep Mode Clock Gating Control Value Description 0 UART module 6 is disabled. 1 Enable and provide a clock to UART module 6 in deep-sleep mode.

Bit/Field	Name	Type	Reset	Description
5	D5	RW	0	UART Module 5 Deep-Sleep Mode Clock Gating Control Value Description 0 UART module 5 is disabled. 1 Enable and provide a clock to UART module 5 in deep-sleep mode.
4	D4	RW	0	UART Module 4 Deep-Sleep Mode Clock Gating Control Value Description 0 UART module 4 is disabled. 1 Enable and provide a clock to UART module 4 in deep-sleep mode.
3	D3	RW	0	UART Module 3 Deep-Sleep Mode Clock Gating Control Value Description 0 UART module 3 is disabled. 1 Enable and provide a clock to UART module 3 in deep-sleep mode.
2	D2	RW	0	UART Module 2 Deep-Sleep Mode Clock Gating Control Value Description 0 UART module 2 is disabled. 1 Enable and provide a clock to UART module 2 in deep-sleep mode.
1	D1	RW	0	UART Module 1 Deep-Sleep Mode Clock Gating Control Value Description 0 UART module 1 is disabled. 1 Enable and provide a clock to UART module 1 in deep-sleep mode.
0	D0	RW	0	UART Module 0 Deep-Sleep Mode Clock Gating Control Value Description 0 UART module 0 is disabled. 1 Enable and provide a clock to UART module 0 in deep-sleep mode.

Register 85: Synchronous Serial Interface Deep-Sleep Mode Clock Gating Control (DCGCSSI), offset 0x81C

The **DCGCSSI** register provides software the capability to enable and disable the SSI modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the SSI modules. To support legacy software, the **DCGC1** register is available. A write to the **DCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **DCGC1** register can be read back correctly with a read of the **DCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as SSI0), the write causes proper operation, but the value of that bit is not reflected in the **DCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Synchronous Serial Interface Deep-Sleep Mode Clock Gating Control (DCGCSSI)

Base 0x400F.E000
Offset 0x81C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												D3	D2	D1	D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	D3	RW	0	SSI Module 3 Deep-Sleep Mode Clock Gating Control Value Description 0 SSI module 3 is disabled. 1 Enable and provide a clock to SSI module 3 in deep-sleep mode.
2	D2	RW	0	SSI Module 2 Deep-Sleep Mode Clock Gating Control Value Description 0 SSI module 2 is disabled. 1 Enable and provide a clock to SSI module 2 in deep-sleep mode.

Bit/Field	Name	Type	Reset	Description
1	D1	RW	0	SSI Module 1 Deep-Sleep Mode Clock Gating Control Value Description 0 SSI module 1 is disabled. 1 Enable and provide a clock to SSI module 1 in deep-sleep mode.
0	D0	RW	0	SSI Module 0 Deep-Sleep Mode Clock Gating Control Value Description 0 SSI module 0 is disabled. 1 Enable and provide a clock to SSI module 0 in deep-sleep mode.

Register 86: Inter-Integrated Circuit Deep-Sleep Mode Clock Gating Control (DCGCI2C), offset 0x820

The **DCGCI2C** register provides software the capability to enable and disable the I^{2}C modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the I^{2}C modules. To support legacy software, the **DCGC1** register is available. A write to the **DCGC1** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **DCGC1** register can be read back correctly with a read of the **DCGC1** register. Software must use this register to support modules that are not present in the legacy registers. If software uses this register to write a legacy peripheral (such as I^{2}C0), the write causes proper operation, but the value of that bit is not reflected in the **DCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Inter-Integrated Circuit Deep-Sleep Mode Clock Gating Control (DCGCI2C)

Base 0x400F.E000
Offset 0x820
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										D5	D4	D3	D2	D1	D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	D5	RW	0	I^{2}C Module 5 Deep-Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 5 is disabled. 1 Enable and provide a clock to I^{2}C module 5 in deep-sleep mode.
4	D4	RW	0	I^{2}C Module 4 Deep-Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 4 is disabled. 1 Enable and provide a clock to I^{2}C module 4 in deep-sleep mode.

Bit/Field	Name	Type	Reset	Description
3	D3	RW	0	I^{2}C Module 3 Deep-Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 3 is disabled. 1 Enable and provide a clock to I^{2}C module 3 in deep-sleep mode.
2	D2	RW	0	I^{2}C Module 2 Deep-Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 2 is disabled. 1 Enable and provide a clock to I^{2}C module 2 in deep-sleep mode.
1	D1	RW	0	I^{2}C Module 1 Deep-Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 1 is disabled. 1 Enable and provide a clock to I^{2}C module 1 in deep-sleep mode.
0	D0	RW	0	I^{2}C Module 0 Deep-Sleep Mode Clock Gating Control Value Description 0 I^{2}C module 0 is disabled. 1 Enable and provide a clock to I^{2}C module 0 in deep-sleep mode.

Register 87: Universal Serial Bus Deep-Sleep Mode Clock Gating Control (DCGCUSB), offset 0x828

The **DCGCUSB** register provides software the capability to enable and disable the USB module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the USB module. To support legacy software, the **DCGC2** register is available. A write to the `USB0` bit in the **DCGC2** register also writes the `D0` bit in this register. If the `USB0` bit is changed by writing to the **DCGC2** register, it can be read back correctly with a read of the **DCGC2** register. If software uses this register to control the clock for the USB module, the write causes proper operation, but the `USB0` bit in the **DCGC2** register does not reflect the value of the `D0` bit. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Universal Serial Bus Deep-Sleep Mode Clock Gating Control (DCGCUSB)

Base 0x400F.E000
Offset 0x828
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	D0	RW	0	USB Module Deep-Sleep Mode Clock Gating Control Value Description 0 USB module is disabled. 1 Enable and provide a clock to the USB module in deep-sleep mode.

Register 88: Controller Area Network Deep-Sleep Mode Clock Gating Control (DCGCCAN), offset 0x834

The **DCGCCAN** register provides software the capability to enable and disable the CAN modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the CAN modules. To support legacy software, the **DCGC0** register is available. A write to the **DCGC0** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **DCGC0** register can be read back correctly with a read of the **DCGC0** register. If software uses this register to write a legacy peripheral (such as CAN0), the write causes proper operation, but the value of that bit is not reflected in the **DCGC0** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Controller Area Network Deep-Sleep Mode Clock Gating Control (DCGCCAN)

Base 0x400F.E000
Offset 0x834
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	D0	RW	0	CAN Module 0 Deep-Sleep Mode Clock Gating Control

Value	Description
0	CAN module 0 is disabled.
1	Enable and provide a clock to CAN module 0 in deep-sleep mode.

Register 89: Analog-to-Digital Converter Deep-Sleep Mode Clock Gating Control (DCGCADC), offset 0x838

The **DCGCADC** register provides software the capability to enable and disable the ADC modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the ADC modules. To support legacy software, the **DCGC0** register is available. A write to the **DCGC0** register also writes the corresponding bit in this register. Any bits that are changed by writing to the **DCGC0** register can be read back correctly with a read of the **DCGC0** register. If software uses this register to write a legacy peripheral (such as ADC0), the write causes proper operation, but the value of that bit is not reflected in the **DCGC0** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Analog-to-Digital Converter Deep-Sleep Mode Clock Gating Control (DCGCADC)

Base 0x400F.E000
Offset 0x838
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														D1	D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	D1	RW	0	ADC Module 1 Deep-Sleep Mode Clock Gating Control Value Description 0 ADC module 1 is disabled. 1 Enable and provide a clock to ADC module 1 in deep-sleep mode.
0	D0	RW	0	ADC Module 0 Deep-Sleep Mode Clock Gating Control Value Description 0 ADC module 0 is disabled. 1 Enable and provide a clock to ADC module 0 in deep-sleep mode.

Register 90: Analog Comparator Deep-Sleep Mode Clock Gating Control (DCGCACMP), offset 0x83C

The **DCGCACMP** register provides software the capability to enable and disable the analog comparator module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the watchdog modules and has the same bit polarity as the corresponding **DCGCn** bits.

Important: This register should be used to control the clocking for the analog comparator module. To support legacy software, the **DCGC1** register is available. Setting any of the `COMPn` bits in the **DCGC1** register also sets the `D0` bit in this register. If any of the `COMPn` bits are set by writing to the **DCGC1** register, it can be read back correctly when reading the **DCGC1** register. If software uses this register to change the clocking for the analog comparator module, the write causes proper operation, but the value `D0` is not reflected by the `COMPn` bits in the **DCGC1** register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Analog Comparator Deep-Sleep Mode Clock Gating Control (DCGCACMP)

Base 0x400F.E000
Offset 0x83C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	D0	RW	0	Analog Comparator Module 0 Deep-Sleep Mode Clock Gating Control Value Description 0 Analog comparator module is disabled. 1 Enable and provide a clock to the analog comparator module in deep-sleep mode.

Register 91: EEPROM Deep-Sleep Mode Clock Gating Control (DCGCEEPROM), offset 0x858

The **DCGCEEPROM** register provides software the capability to enable and disable the EEPROM module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

EEPROM Deep-Sleep Mode Clock Gating Control (DCGCEEPROM)

Base 0x400F.E000
Offset 0x858
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	D0	RW	0	EEPROM Module Deep-Sleep Mode Clock Gating Control

Value	Description
0	EEPROM module is disabled.
1	Enable and provide a clock to the EEPROM module in deep-sleep mode.

Register 92: 32/64-Bit Wide General-Purpose Timer Deep-Sleep Mode Clock Gating Control (DCGCWTIMER), offset 0x85C

The **DCGCWTIMER** register provides software the capability to enable and disable 32/64-bit wide timer modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power. This register provides the same capability as the legacy **Deep-Sleep Mode Clock Gating Control Register n DCGCn** registers specifically for the timer modules and has the same bit polarity as the corresponding **DCGCn** bits.

32/64-Bit Wide General-Purpose Timer Deep-Sleep Mode Clock Gating Control (DCGCWTIMER)

Base 0x400F.E000
Offset 0x85C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										D5	D4	D3	D2	D1	D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	D5	RW	0	32/64-Bit Wide General-Purpose Timer 5 Deep-Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 5 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 5 in deep-sleep mode.
4	D4	RW	0	32/64-Bit Wide General-Purpose Timer 4 Deep-Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 4 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 4 in deep-sleep mode.
3	D3	RW	0	32/64-Bit Wide General-Purpose Timer 3 Deep-Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 3 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 3 in deep-sleep mode.

Bit/Field	Name	Type	Reset	Description
2	D2	RW	0	32/64-Bit Wide General-Purpose Timer 2 Deep-Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 2 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 2 in deep-sleep mode.
1	D1	RW	0	32/64-Bit Wide General-Purpose Timer 1 Deep-Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 1 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 1 in deep-sleep mode.
0	D0	RW	0	32/64-Bit Wide General-Purpose Timer 0 Deep-Sleep Mode Clock Gating Control Value Description 0 32/64-bit wide general-purpose timer module 0 is disabled. 1 Enable and provide a clock to 32/64-bit wide general-purpose timer module 0 in deep-sleep mode.

Register 93: Watchdog Timer Peripheral Ready (PRWD), offset 0xA00

The **PRWD** register indicates whether the watchdog modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCWD** bit is changed. A reset change is initiated if the corresponding **SRWD** bit is changed from 0 to 1.

The **PRWD** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Watchdog Timer Peripheral Ready (PRWD)

Base 0x400F.E000
Offset 0xA00
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	R1	RO	0	Watchdog Timer 1 Peripheral Ready Value Description 0 Watchdog module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 Watchdog module 1 is ready for access.
0	R0	RO	0	Watchdog Timer 0 Peripheral Ready Value Description 0 Watchdog module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 Watchdog module 0 is ready for access.

Register 94: 16/32-Bit General-Purpose Timer Peripheral Ready (PRTIMER), offset 0xA04

The **PRTIMER** register indicates whether the timer modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCTIMER** bit is changed. A reset change is initiated if the corresponding **SRTIMER** bit is changed from 0 to 1.

The **PRTIMER** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

16/32-Bit General-Purpose Timer Peripheral Ready (PRTIMER)

Base 0x400F.E000
Offset 0xA04
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	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	R5	RO	0	16/32-Bit General-Purpose Timer 5 Peripheral Ready Value Description 0 16/32-bit timer module 5 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 16/32-bit timer module 5 is ready for access.
4	R4	RO	0	16/32-Bit General-Purpose Timer 4 Peripheral Ready Value Description 0 16/32-bit timer module 4 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 16/32-bit timer module 4 is ready for access.
3	R3	RO	0	16/32-Bit General-Purpose Timer 3 Peripheral Ready Value Description 0 16/32-bit timer module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 16/32-bit timer module 3 is ready for access.

Bit/Field	Name	Type	Reset	Description
2	R2	RO	0	16/32-Bit General-Purpose Timer 2 Peripheral Ready Value Description 0 16/32-bit timer module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 16/32-bit timer module 2 is ready for access.
1	R1	RO	0	16/32-Bit General-Purpose Timer 1 Peripheral Ready Value Description 0 16/32-bit timer module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 16/32-bit timer module 1 is ready for access.
0	R0	RO	0	16/32-Bit General-Purpose Timer 0 Peripheral Ready Value Description 0 16/32-bit timer module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 16/32-bit timer module 0 is ready for access.

Register 95: General-Purpose Input/Output Peripheral Ready (PRGPIO), offset 0xA08

The **PRGPIO** register indicates whether the GPIO modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCGPIO** bit is changed. A reset change is initiated if the corresponding **SRGPIO** bit is changed from 0 to 1.

The **PRGPIO** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

General-Purpose Input/Output Peripheral Ready (PRGPIO)

Base 0x400F.E000
Offset 0xA08
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									R6	R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	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	R6	RO	0	GPIO Port G Peripheral Ready Value Description 0 GPIO Port G is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 GPIO Port G is ready for access.
5	R5	RO	0	GPIO Port F Peripheral Ready Value Description 0 GPIO Port F is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 GPIO Port F is ready for access.
4	R4	RO	0	GPIO Port E Peripheral Ready Value Description 0 GPIO Port E is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 GPIO Port E is ready for access.

Bit/Field	Name	Type	Reset	Description
3	R3	RO	0	GPIO Port D Peripheral Ready Value Description 0 GPIO Port D is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 GPIO Port D is ready for access.
2	R2	RO	0	GPIO Port C Peripheral Ready Value Description 0 GPIO Port C is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 GPIO Port C is ready for access.
1	R1	RO	0	GPIO Port B Peripheral Ready Value Description 0 GPIO Port B is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 GPIO Port B is ready for access.
0	R0	RO	0	GPIO Port A Peripheral Ready Value Description 0 GPIO Port A is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 GPIO Port A is ready for access.

Register 96: Micro Direct Memory Access Peripheral Ready (PRDMA), offset 0xA0C

The **PRDMA** register indicates whether the µDMA module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCDMA** bit is changed. A reset change is initiated if the corresponding **SRDMA** bit is changed from 0 to 1.

The **PRDMA** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Micro Direct Memory Access Peripheral Ready (PRDMA)

Base 0x400F.E000
Offset 0xA0C
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RO	0	µDMA Module Peripheral Ready Value Description 0 The µDMA module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 The µDMA module is ready for access.

Register 97: Universal Asynchronous Receiver/Transmitter Peripheral Ready (PRUART), offset 0xA18

The **PRUART** register indicates whether the UART modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCUART** bit is changed. A reset change is initiated if the corresponding **SRUART** bit is changed from 0 to 1.

The **PRUART** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Universal Asynchronous Receiver/Transmitter Peripheral Ready (PRUART)

Base 0x400F.E000
Offset 0xA18
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								R7	R6	R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	R7	RO	0	UART Module 7 Peripheral Ready Value Description 0 UART module 7 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 UART module 7 is ready for access.
6	R6	RO	0	UART Module 6 Peripheral Ready Value Description 0 UART module 6 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 UART module 6 is ready for access.
5	R5	RO	0	UART Module 5 Peripheral Ready Value Description 0 UART module 5 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 UART module 5 is ready for access.

Bit/Field	Name	Type	Reset	Description
4	R4	RO	0	UART Module 4 Peripheral Ready Value Description 0 UART module 4 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 UART module 4 is ready for access.
3	R3	RO	0	UART Module 3 Peripheral Ready Value Description 0 UART module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 UART module 3 is ready for access.
2	R2	RO	0	UART Module 2 Peripheral Ready Value Description 0 UART module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 UART module 2 is ready for access.
1	R1	RO	0	UART Module 1 Peripheral Ready Value Description 0 UART module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 UART module 1 is ready for access.
0	R0	RO	0	UART Module 0 Peripheral Ready Value Description 0 UART module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 UART module 0 is ready for access.

Register 98: Synchronous Serial Interface Peripheral Ready (PRSSI), offset 0xA1C

The **PRSSI** register indicates whether the SSI modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCSSI** bit is changed. A reset change is initiated if the corresponding **SRSSI** bit is changed from 0 to 1.

The **PRSSI** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Synchronous Serial Interface Peripheral Ready (PRSSI)

Base 0x400F.E000
Offset 0xA1C
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	R3	RO	0	SSI Module 3 Peripheral Ready Value Description 0 SSI module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 SSI module 3 is ready for access.
2	R2	RO	0	SSI Module 2 Peripheral Ready Value Description 0 SSI module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 SSI module 2 is ready for access.
1	R1	RO	0	SSI Module 1 Peripheral Ready Value Description 0 SSI module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 SSI module 1 is ready for access.

Bit/Field	Name	Type	Reset	Description
0	R0	RO	0	SSI Module 0 Peripheral Ready Value Description 0 SSI module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 SSI module 0 is ready for access.

Register 99: Inter-Integrated Circuit Peripheral Ready (PRI2C), offset 0xA20

The **PRI2C** register indicates whether the I^{2}C modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCI2C** bit is changed. A reset change is initiated if the corresponding **SRI2C** bit is changed from 0 to 1.

The **PRI2C** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Inter-Integrated Circuit Peripheral Ready (PRI2C)

Base 0x400F.E000
Offset 0xA20
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	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	R5	RO	0	I^{2}C Module 5 Peripheral Ready Value Description 0 I^{2}C module 5 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 I^{2}C module 5 is ready for access.
4	R4	RO	0	I^{2}C Module 4 Peripheral Ready Value Description 0 I^{2}C module 4 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 I^{2}C module 4 is ready for access.
3	R3	RO	0	I^{2}C Module 3 Peripheral Ready Value Description 0 I^{2}C module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 I^{2}C module 3 is ready for access.

Bit/Field	Name	Type	Reset	Description
2	R2	RO	0	I^2C Module 2 Peripheral Ready Value Description 0 I^2C module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 I^2C module 2 is ready for access.
1	R1	RO	0	I^2C Module 1 Peripheral Ready Value Description 0 I^2C module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 I^2C module 1 is ready for access.
0	R0	RO	0	I^2C Module 0 Peripheral Ready Value Description 0 I^2C module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 I^2C module 0 is ready for access.

Register 100: Universal Serial Bus Peripheral Ready (PRUSB), offset 0xA28

The **PRUSB** register indicates whether the USB module is ready to be accessed by software following a change in Run mode clocking or reset. A Run mode clocking change is initiated if the corresponding **RCGCUSB** bit is changed. A reset change is initiated if the corresponding **SRUSB** bit is changed from 0 to 1.

The **PRUSB** bit is cleared on either of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Universal Serial Bus Peripheral Ready (PRUSB)

Base 0x400F.E000
Offset 0xA28
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RO	0	USB Module Peripheral Ready

Value	Description
0	The USB module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.
1	The USB module is ready for access.

Register 101: Controller Area Network Peripheral Ready (PRCAN), offset 0xA34

The **PRCAN** register indicates whether the CAN modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCCAN** bit is changed. A reset change is initiated if the corresponding **SRCAN** bit is changed from 0 to 1.

The **PRCAN** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Controller Area Network Peripheral Ready (PRCAN)

Base 0x400F.E000
Offset 0xA34
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RO	0	CAN Module 0 Peripheral Ready

Value	Description
0	CAN module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.
1	CAN module 0 is ready for access.

Register 102: Analog-to-Digital Converter Peripheral Ready (PRADC), offset 0xA38

The **PRADC** register indicates whether the ADC modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCADC** bit is changed. A reset change is initiated if the corresponding **SRADC** bit is changed from 0 to 1.

The **PRADC** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Analog-to-Digital Converter Peripheral Ready (PRADC)

Base 0x400F.E000
Offset 0xA38
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	R1	RO	0	ADC Module 1 Peripheral Ready Value Description 0 ADC module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 ADC module 1 is ready for access.
0	R0	RO	0	ADC Module 0 Peripheral Ready Value Description 0 ADC module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 ADC module 0 is ready for access.

Register 103: Analog Comparator Peripheral Ready (PRACMP), offset 0xA3C

The **PRACMP** register indicates whether the analog comparator module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCACMP** bit is changed. A reset change is initiated if the corresponding **SRACMP** bit is changed from 0 to 1.

The **PRACMP** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Analog Comparator Peripheral Ready (PRACMP)

Base 0x400F.E000
Offset 0xA3C
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RO	0	Analog Comparator Module 0 Peripheral Ready

Value	Description
0	The analog comparator module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.
1	The analog comparator module is ready for access.

Register 104: EEPROM Peripheral Ready (PREEPROM), offset 0xA58

The **PREEPROM** register indicates whether the EEPROM module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCEEPROM** bit is changed. A reset change is initiated if the corresponding **SREEPROM** bit is changed from 0 to 1.

The **PREEPROM** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

EEPROM Peripheral Ready (PREEPROM)

Base 0x400F.E000
Offset 0xA58
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	R0	RO	0	EEPROM Module Peripheral Ready

Value	Description
0	The EEPROM module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.
1	The EEPROM module is ready for access.

Register 105: 32/64-Bit Wide General-Purpose Timer Peripheral Ready (PRWTIMER), offset 0xA5C

The **PRWTIMER** register indicates whether the timer modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCWTIMER** bit is changed. A reset change is initiated if the corresponding **SRWTIMER** bit is changed from 0 to 1.

The **PRWTIMER** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

32/64-Bit Wide General-Purpose Timer Peripheral Ready (PRWTIMER)

Base 0x400F.E000
Offset 0xA5C
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										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	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	R5	RO	0	32/64-Bit Wide General-Purpose Timer 5 Peripheral Ready Value Description 0 32/64-bit wide timer module 5 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 32/64-bit wide timer module 5 is ready for access.
4	R4	RO	0	32/64-Bit Wide General-Purpose Timer 4 Peripheral Ready Value Description 0 32/64-bit wide timer module 4 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 32/64-bit wide timer module 4 is ready for access.
3	R3	RO	0	32/64-Bit Wide General-Purpose Timer 3 Peripheral Ready Value Description 0 32/64-bit wide timer module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 32/64-bit wide timer module 3 is ready for access.

Bit/Field	Name	Type	Reset	Description
2	R2	RO	0	32/64-Bit Wide General-Purpose Timer 2 Peripheral Ready Value Description 0 32/64-bit wide timer module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 32/64-bit wide timer module 2 is ready for access.
1	R1	RO	0	32/64-Bit Wide General-Purpose Timer 1 Peripheral Ready Value Description 0 32/64-bit wide timer module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 32/64-bit wide timer module 1 is ready for access.
0	R0	RO	0	32/64-Bit Wide General-Purpose Timer 0 Peripheral Ready Value Description 0 32/64-bit wide timer module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. 1 32/64-bit wide timer module 0 is ready for access.

5.6 System Control Legacy Register Descriptions

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

Important: Register in this section are provided for legacy software support only; registers in “System Control Register Descriptions” on page 224 should be used instead.

Register 106: Device Capabilities 0 (DC0), offset 0x008

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

Important: This register is provided for legacy software support only.

The **Flash Size (FSIZE)** and **SRAM Size (SSIZE)** registers should be used to determine this microcontroller's memory sizes. A read of **DC0** correctly identifies legacy memory sizes but software must use **FSIZE** and **SSIZE** for memory sizes that are not listed below.

Device Capabilities 0 (DC0)

Base 0x400F.E000
Offset 0x008
Type RO, reset 0x002F.000F

	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	0	0	0	1	0	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	0	0	0	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:16	SRAMSZ	RO	0x2F	SRAM Size Indicates the size of the on-chip SRAM.

Value	Description
0x7	2 KB of SRAM
0xF	4 KB of SRAM
0x17	6 KB of SRAM
0x1F	8 KB of SRAM
0x2F	12 KB of SRAM
0x3F	16 KB of SRAM
0x4F	20 KB of SRAM
0x5F	24 KB of SRAM
0x7F	32 KB of SRAM

Bit/Field	Name	Type	Reset	Description
15:0	FLASHSZ	RO	0xF	Flash Size Indicates the size of the on-chip Flash memory.

Value	Description
0x3	8 KB of Flash
0x7	16 KB of Flash
0xF	32 KB of Flash
0x1F	64 KB of Flash
0x2F	96 KB of Flash
0x3F	128 KB of Flash
0x5F	192 KB of Flash
0x7F	256 KB of Flash

Register 107: 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**, **DCGC0**, and the peripheral-specific **RCGC**, **SCGC**, and **DCGC** registers cannot be set.

Important: This register is provided for legacy software support only.

The Peripheral Present registers should be used to determine which modules are implemented on this microcontroller. A read of **DC1** correctly identifies if a legacy module is present but software must use the Peripheral Present registers to determine if a module is present that is not supported by the **DCn** registers.

Likewise, the **ADC Peripheral Properties (ADCPP)** register should be used to determine the maximum ADC sample rate and whether the temperature sensor is present. However, to support legacy software, the MAXADCnSPD fields and the TEMPSNS bit are available. A read of **DC1** correctly identifies the maximum ADC sample rate for legacy rates and whether the temperature sensor is present.

Device Capabilities 1 (DC1)

Base 0x400F.E000
Offset 0x010
Type RO, reset 0x1103.2FBF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved		CAN1	CAN0	reserved		PWM1	PWM0	reserved		ADC1	ADC0
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	1	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	0	0	1	0	1	1	1	1	1	0	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	0x1	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	0x0	CAN Module 1 Present When set, indicates that CAN unit 1 is present.
24	CAN0	RO	0x1	CAN Module 0 Present When set, indicates that CAN unit 0 is present.
23:22	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
21	PWM1	RO	0x0	PWM Module 1 Present When set, indicates that the PWM module is present.

Bit/Field	Name	Type	Reset	Description
20	PWM0	RO	0x0	PWM Module 0 Present When set, indicates that the PWM module is present.
19:18	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
17	ADC1	RO	0x1	ADC Module 1 Present When set, indicates that ADC module 1 is present.
16	ADC0	RO	0x1	ADC Module 0 Present When set, indicates that ADC module 0 is present
15:12	MINSYSDIV	RO	0x2	System Clock Divider Minimum 4-bit divider value for system clock. The reset value is hardware-dependent. See the **RCC** register for how to change the system clock divisor using the `SYSDIV` bit. Value Description 0x1 Reserved 0x2 Specifies an 80-MHz CPU clock with a PLL divider of 2.5. 0x3 Specifies a 50-MHz CPU clock with a PLL divider of 4. 0x4 Specifies a 40-MHz CPU clock with a PLL divider of 5. 0x7 Specifies a 25-MHz clock with a PLL divider of 8. 0x9 Specifies a 20-MHz clock with a PLL divider of 10.
11:10	MAXADC1SPD	RO	0x3	Max ADC1 Speed This field indicates the maximum rate at which the ADC samples data. Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 0x0 125K samples/second
9:8	MAXADC0SPD	RO	0x3	Max ADC0 Speed This field indicates the maximum rate at which the ADC samples data. Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 0x0 125K samples/second
7	MPU	RO	0x1	MPU Present When set, indicates that the Cortex-M4F Memory Protection Unit (MPU) module is present. See the "Cortex-M4F Peripherals" chapter for details on the MPU.

Bit/Field	Name	Type	Reset	Description
6	HIB	RO	0x0	Hibernation Module Present When set, indicates that the Hibernation module is present.
5	TEMPSNS	RO	0x1	Temp Sensor Present When set, indicates that the on-chip temperature sensor is present.
4	PLL	RO	0x1	PLL Present When set, indicates that the on-chip Phase Locked Loop (PLL) is present.
3	WDT0	RO	0x1	Watchdog Timer 0 Present When set, indicates that watchdog timer 0 is present.
2	SWO	RO	0x1	SWO Trace Port Present When set, indicates that the Serial Wire Output (SWO) trace port is present.
1	SWD	RO	0x1	SWD Present When set, indicates that the Serial Wire Debugger (SWD) is present.
0	JTAG	RO	0x1	JTAG Present When set, indicates that the JTAG debugger interface is present.

Register 108: 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 **RCGC1**, **SCGC1**, **DCGC1**, and the peripheral-specific **RCGC**, **SCGC**, and **DCGC** registers registers cannot be set.

Important: This register is provided for legacy software support only.

The Peripheral Present registers should be used to determine which modules are implemented on this microcontroller. A read of **DC2** correctly identifies if a legacy module is present but software must use the Peripheral Present registers to determine if a module is present that is not supported by the **DCn** registers.

Note that the **Analog Comparator Peripheral Present (PPACMP)** register identifies whether the analog comparator module is present. The **Analog Comparator Peripheral Properties (ACMPPP)** register indicates how many analog comparator blocks are present in the module.

Device Capabilities 2 (DC2)

Base 0x400F.E000
Offset 0x014
Type RO, reset 0x030F.F037

	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	0	0	0	0	0	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
	I2C1HS	I2C1	I2C0HS	I2C0	reserved		QEI1	QEI0	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	1	1	1	1	0	0	0	0	0	0	1	1	0	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	EPI0	RO	0x0	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	0x0	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	0x0	Analog Comparator 2 Present When set, indicates that analog comparator 2 is present.
25	COMP1	RO	0x1	Analog Comparator 1 Present When set, indicates that analog comparator 1 is present.

Bit/Field	Name	Type	Reset	Description
24	COMP0	RO	0x1	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	0x1	Timer Module 3 Present When set, indicates that General-Purpose Timer module 3 is present.
18	TIMER2	RO	0x1	Timer Module 2 Present When set, indicates that General-Purpose Timer module 2 is present.
17	TIMER1	RO	0x1	Timer Module 1 Present When set, indicates that General-Purpose Timer module 1 is present.
16	TIMER0	RO	0x1	Timer Module 0 Present When set, indicates that General-Purpose Timer module 0 is present.
15	I2C1HS	RO	0x1	I2C Module 1 Speed When set, indicates that I2C module 1 can operate in high-speed mode.
14	I2C1	RO	0x1	I2C Module 1 Present When set, indicates that I2C module 1 is present.
13	I2C0HS	RO	0x1	I2C Module 0 Speed When set, indicates that I2C module 0 can operate in high-speed mode.
12	I2C0	RO	0x1	I2C Module 0 Present When set, indicates that I2C module 0 is present.
11:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	QEI1	RO	0x0	QEI Module 1 Present When set, indicates that QEI module 1 is present.
8	QEI0	RO	0x0	QEI Module 0 Present When set, indicates that QEI module 0 is present.
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	RO	0x1	SSI Module 1 Present When set, indicates that SSI module 1 is present.
4	SSI0	RO	0x1	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.

Bit/Field	Name	Type	Reset	Description
2	UART2	RO	0x1	UART Module 2 Present When set, indicates that UART module 2 is present.
1	UART1	RO	0x1	UART Module 1 Present When set, indicates that UART module 1 is present.
0	UART0	RO	0x1	UART Module 0 Present When set, indicates that UART module 0 is present.

Register 109: 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 feature is not present.

Important: This register is provided for legacy software support only.

For some modules, the peripheral-resident Peripheral Properties registers should be used to determine which pins are available on this microcontroller. A read of **DC3** correctly identifies if a legacy pin is present but software must use the Peripheral Properties registers to determine if a pin is present that is not supported by the **DCn** registers.

Device Capabilities 3 (DC3)

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

	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
	PWMFAULT	C2O	C2PLUS	C2MINUS	C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	32KHZ	RO	0x1	32KHz Input Clock Available When set, indicates an even CCP pin is present and can be used as a 32-KHz input clock. **Note:** The **GPTMPP** register does not provide this information.
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	0x1	T2CCP1 Pin Present When set, indicates that Capture/Compare/PWM pin `T2CCP1` is present. **Note:** The **GPTMPP** register does not provide this information.
28	CCP4	RO	0x1	T2CCP0 Pin Present When set, indicates that Capture/Compare/PWM pin `T2CCP0` is present. **Note:** The **GPTMPP** register does not provide this information.
27	CCP3	RO	0x1	T1CCP1 Pin Present When set, indicates that Capture/Compare/PWM pin `T1CCP1` is present. **Note:** The **GPTMPP** register does not provide this information.
26	CCP2	RO	0x1	T1CCP0 Pin Present When set, indicates that Capture/Compare/PWM pin `T1CCP0` is present. **Note:** The **GPTMPP** register does not provide this information.

Bit/Field	Name	Type	Reset	Description
25	CCP1	RO	0x1	T0CCP1 Pin Present When set, indicates that Capture/Compare/PWM pin T0CCP1 is present. **Note:** The **GPTMPP** register does not provide this information.
24	CCP0	RO	0x1	T0CCP0 Pin Present When set, indicates that Capture/Compare/PWM pin T0CCP0 is present. **Note:** The **GPTMPP** register does not provide this information.
23	ADC0AIN7	RO	0x1	ADC Module 0 AIN7 Pin Present When set, indicates that ADC module 0 input pin 7 is present. **Note:** The CH field in the **ADCPP** register provides this information.
22	ADC0AIN6	RO	0x1	ADC Module 0 AIN6 Pin Present When set, indicates that ADC module 0 input pin 6 is present. **Note:** The CH field in the **ADCPP** register provides this information.
21	ADC0AIN5	RO	0x1	ADC Module 0 AIN5 Pin Present When set, indicates that ADC module 0 input pin 5 is present. **Note:** The CH field in the **ADCPP** register provides this information.
20	ADC0AIN4	RO	0x1	ADC Module 0 AIN4 Pin Present When set, indicates that ADC module 0 input pin 4 is present. **Note:** The CH field in the **ADCPP** register provides this information.
19	ADC0AIN3	RO	0x1	ADC Module 0 AIN3 Pin Present When set, indicates that ADC module 0 input pin 3 is present. **Note:** The CH field in the **ADCPP** register provides this information.
18	ADC0AIN2	RO	0x1	ADC Module 0 AIN2 Pin Present When set, indicates that ADC module 0 input pin 2 is present. **Note:** The CH field in the **ADCPP** register provides this information.
17	ADC0AIN1	RO	0x1	ADC Module 0 AIN1 Pin Present When set, indicates that ADC module 0 input pin 1 is present. **Note:** The CH field in the **ADCPP** register provides this information.
16	ADC0AIN0	RO	0x1	ADC Module 0 AIN0 Pin Present When set, indicates that ADC module 0 input pin 0 is present. **Note:** The CH field in the **ADCPP** register provides this information.
15	PWMFAULT	RO	0x0	PWM Fault Pin Present When set, indicates that a PWM Fault pin is present. See **DC5** for specific Fault pins on this device. **Note:** The FCNT field in the **PWMPP** register provides this information.
14	C2O	RO	0x0	C2o Pin Present When set, indicates that the analog comparator 2 output pin is present. **Note:** The C2O bit in the **ACMPPP** register provides this information.

Bit/Field	Name	Type	Reset	Description
13	C2PLUS	RO	0x0	C2+ Pin Present When set, indicates that the analog comparator 2 (+) input pin is present. **Note:** This pin is present when analog comparator 2 is present.
12	C2MINUS	RO	0x0	C2- Pin Present When set, indicates that the analog comparator 2 (-) input pin is present. **Note:** This pin is present when analog comparator 2 is present.
11	C1O	RO	0x1	C1o Pin Present When set, indicates that the analog comparator 1 output pin is present. **Note:** The `C1O` bit in the **ACMPPP** register provides this information.
10	C1PLUS	RO	0x1	C1+ Pin Present When set, indicates that the analog comparator 1 (+) input pin is present. **Note:** This pin is present when analog comparator 1 is present.
9	C1MINUS	RO	0x1	C1- Pin Present When set, indicates that the analog comparator 1 (-) input pin is present. **Note:** This pin is present when analog comparator 1 is present.
8	C0O	RO	0x1	C0o Pin Present When set, indicates that the analog comparator 0 output pin is present. **Note:** The `C0O` bit in the **ACMPPP** register provides this information.
7	C0PLUS	RO	0x1	C0+ Pin Present When set, indicates that the analog comparator 0 (+) input pin is present. **Note:** This pin is present when analog comparator 0 is present.
6	C0MINUS	RO	0x1	C0- Pin Present When set, indicates that the analog comparator 0 (-) input pin is present. **Note:** This pin is present when analog comparator 0 is present.
5	PWM5	RO	0x0	PWM5 Pin Present When set, indicates that the PWM pin 5 is present. **Note:** The `GCNT` field in the **PWMPP** register provides this information.
4	PWM4	RO	0x0	PWM4 Pin Present When set, indicates that the PWM pin 4 is present. **Note:** The `GCNT` field in the **PWMPP** register provides this information.
3	PWM3	RO	0x0	PWM3 Pin Present When set, indicates that the PWM pin 3 is present. **Note:** The `GCNT` field in the **PWMPP** register provides this information.

Bit/Field	Name	Type	Reset	Description
2	PWM2	RO	0x0	PWM2 Pin Present When set, indicates that the PWM pin 2 is present. **Note:** The `GCNT` field in the **PWMPP** register provides this information.
1	PWM1	RO	0x0	PWM1 Pin Present When set, indicates that the PWM pin 1 is present. **Note:** The `GCNT` field in the **PWMPP** register provides this information.
0	PWM0	RO	0x0	PWM0 Pin Present When set, indicates that the PWM pin 0 is present. **Note:** The `GCNT` field in the **PWMPP** register provides this information.

Register 110: 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 **RCGC2**, **SCGC2**, **DCGC2**, and the peripheral-specific **RCGC**, **SCGC**, and **DCGC** registers registers cannot be set.

Important: This register is provided for legacy software support only.

The Peripheral Present registers should be used to determine which modules are implemented on this microcontroller. A read of **DC4** correctly identifies if a legacy module is present but software must use the Peripheral Present registers to determine if a module is present that is not supported by the **DCn** registers.

The peripheral-resident Peripheral Properties registers should be used to determine which pins and features are available on this microcontroller. A read of **DC4** correctly identifies if a legacy pin or feature is present. Software must use the Peripheral Properties registers to determine if a pin or feature is present that is not supported by the **DCn** registers.

Device Capabilities 4 (DC4)

Base 0x400F.E000
Offset 0x01C
Type RO, reset 0x0004.F07F

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

	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	GPIOD	GPIOC	GPIOB	GPIOA
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	0	0	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	RO	0x0	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	0x0	Ethernet MAC Layer 0 Present When set, indicates that Ethernet MAC layer 0 is present.
27:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	E1588	RO	0x0	1588 Capable When set, indicates that Ethernet MAC layer 0 is 1588 capable.

Bit/Field	Name	Type	Reset	Description
23:19	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	0x1	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	0x1	T3CCP1 Pin Present When set, indicates that Capture/Compare/PWM pin `T3CCP1` is present. **Note:** The **GPTMPP** register does not provide this information.
14	CCP6	RO	0x1	T3CCP0 Pin Present When set, indicates that Capture/Compare/PWM pin `T3CCP0` is present. **Note:** The **GPTMPP** register does not provide this information.
13	UDMA	RO	0x1	Micro-DMA Module Present When set, indicates that the micro-DMA module present.
12	ROM	RO	0x1	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	0x0	GPIO Port J Present When set, indicates that GPIO Port J is present.
7	GPIOH	RO	0x0	GPIO Port H Present When set, indicates that GPIO Port H is present.
6	GPIOG	RO	0x1	GPIO Port G Present When set, indicates that GPIO Port G is present.
5	GPIOF	RO	0x1	GPIO Port F Present When set, indicates that GPIO Port F is present.
4	GPIOE	RO	0x1	GPIO Port E Present When set, indicates that GPIO Port E is present.
3	GPIOD	RO	0x1	GPIO Port D Present When set, indicates that GPIO Port D is present.
2	GPIOC	RO	0x1	GPIO Port C Present When set, indicates that GPIO Port C is present.
1	GPIOB	RO	0x1	GPIO Port B Present When set, indicates that GPIO Port B is present.

Bit/Field	Name	Type	Reset	Description
0	GPIOA	RO	0x1	GPIO Port A Present When set, indicates that GPIO Port A is present.

Register 111: Device Capabilities 5 (DC5), offset 0x020

This register is predefined by the part and can be used to verify PWM features. If any bit is clear in this register, the module is not present.

Important: This register is provided for legacy software support only.

The **PWM Peripheral Properties (PWMPP)** register should be used to determine what pins and features are available on PWM modules. A read of this register correctly identifies if a legacy pin or feature is present. Software must use the **PWMPP** register to determine if a pin or feature that is not supported by the **DCn** registers is present.

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				PWMFAULT3	PWMFAULT2	PWMFAULT1	PWMFAULT0	reserved		PWMEFLT	PWMESYNC	reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:28	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27	PWMFAULT3	RO	0x0	PWM Fault 3 Pin Present When set, indicates that the PWM Fault 3 pin is present.
26	PWMFAULT2	RO	0x0	PWM Fault 2 Pin Present When set, indicates that the PWM Fault 2 pin is present.
25	PWMFAULT1	RO	0x0	PWM Fault 1 Pin Present When set, indicates that the PWM Fault 1 pin is present.
24	PWMFAULT0	RO	0x0	PWM Fault 0 Pin Present When set, indicates that the PWM Fault 0 pin is present.
23:22	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
21	PWMEFLT	RO	0x0	PWM Extended Fault Active When set, indicates that the PWM Extended Fault feature is active.
20	PWMESYNC	RO	0x0	PWM Extended SYNC Active When set, indicates that the PWM Extended SYNC feature is active.
19:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7	PWM7	RO	0x0	PWM7 Pin Present When set, indicates that the PWM pin 7 is present.
6	PWM6	RO	0x0	PWM6 Pin Present When set, indicates that the PWM pin 6 is present.
5	PWM5	RO	0x0	PWM5 Pin Present When set, indicates that the PWM pin 5 is present.
4	PWM4	RO	0x0	PWM4 Pin Present When set, indicates that the PWM pin 4 is present.
3	PWM3	RO	0x0	PWM3 Pin Present When set, indicates that the PWM pin 3 is present.
2	PWM2	RO	0x0	PWM2 Pin Present When set, indicates that the PWM pin 2 is present.
1	PWM1	RO	0x0	PWM1 Pin Present When set, indicates that the PWM pin 1 is present.
0	PWM0	RO	0x0	PWM0 Pin Present When set, indicates that the PWM pin 0 is present.

Register 112: 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.

Important: This register is provided for legacy software support only.

The **USB Peripheral Properties (USBPP)** register should be used to determine what features are available on the USB module. A read of this register correctly identifies if a legacy feature is present. Software must use the **USBPP** register to determine if a pin or feature that is not supported by the **DCn** registers is present.

Device Capabilities 6 (DC6)

Base 0x400F.E000
Offset 0x024
Type RO, reset 0x0000.0011

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0	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	0x1	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	0x1	USB Module 0 Present This field indicates that USB module 0 is present and specifies its capability.

sysValue	Description
0x0	NA USB0 is not present.
0x1	DEVICE USB0 is Device Only.
0x2	HOST USB0 is Device or Host.
0x3	OTG USB0 is OTG.

Register 113: Device Capabilities 7 (DC7), offset 0x028

This register is predefined by the part and can be used to verify μDMA 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. Channels can have multiple assignments, see “Channel Assignments” on page 519 for more information.

Important: This register is provided for legacy software support only. The DMACHANS bit field in the **DMA Status (DMASTAT)** register indicates the number of DMA channels.

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	0x1	DMA Channel 31 When set, indicates μDMA channel 31 is available.
30	DMACH30	RO	0x1	DMA Channel 30 When set, indicates μDMA channel 30 is available.
29	DMACH29	RO	0x1	DMA Channel 29 When set, indicates μDMA channel 29 is available.
28	DMACH28	RO	0x1	DMA Channel 28 When set, indicates μDMA channel 28 is available.
27	DMACH27	RO	0x1	DMA Channel 27 When set, indicates μDMA channel 27 is available.
26	DMACH26	RO	0x1	DMA Channel 26 When set, indicates μDMA channel 26 is available.
25	DMACH25	RO	0x1	DMA Channel 25 When set, indicates μDMA channel 25 is available.
24	DMACH24	RO	0x1	DMA Channel 24 When set, indicates μDMA channel 24 is available.
23	DMACH23	RO	0x1	DMA Channel 23 When set, indicates μDMA channel 23 is available.
22	DMACH22	RO	0x1	DMA Channel 22 When set, indicates μDMA channel 22 is available.

Bit/Field	Name	Type	Reset	Description
21	DMACH21	RO	0x1	DMA Channel 21 When set, indicates µDMA channel 21 is available.
20	DMACH20	RO	0x1	DMA Channel 20 When set, indicates µDMA channel 20 is available.
19	DMACH19	RO	0x1	DMA Channel 19 When set, indicates µDMA channel 19 is available.
18	DMACH18	RO	0x1	DMA Channel 18 When set, indicates µDMA channel 18 is available.
17	DMACH17	RO	0x1	DMA Channel 17 When set, indicates µDMA channel 17 is available.
16	DMACH16	RO	0x1	DMA Channel 16 When set, indicates µDMA channel 16 is available.
15	DMACH15	RO	0x1	DMA Channel 15 When set, indicates µDMA channel 15 is available.
14	DMACH14	RO	0x1	DMA Channel 14 When set, indicates µDMA channel 14 is available.
13	DMACH13	RO	0x1	DMA Channel 13 When set, indicates µDMA channel 13 is available.
12	DMACH12	RO	0x1	DMA Channel 12 When set, indicates µDMA channel 12 is available.
11	DMACH11	RO	0x1	DMA Channel 11 When set, indicates µDMA channel 11 is available.
10	DMACH10	RO	0x1	DMA Channel 10 When set, indicates µDMA channel 10 is available.
9	DMACH9	RO	0x1	DMA Channel 9 When set, indicates µDMA channel 9 is available.
8	DMACH8	RO	0x1	DMA Channel 8 When set, indicates µDMA channel 8 is available.
7	DMACH7	RO	0x1	DMA Channel 7 When set, indicates µDMA channel 7 is available.
6	DMACH6	RO	0x1	DMA Channel 6 When set, indicates µDMA channel 6 is available.
5	DMACH5	RO	0x1	DMA Channel 5 When set, indicates µDMA channel 5 is available.
4	DMACH4	RO	0x1	DMA Channel 4 When set, indicates µDMA channel 4 is available.

Bit/Field	Name	Type	Reset	Description
3	DMACH3	RO	0x1	DMA Channel 3 When set, indicates μDMA channel 3 is available.
2	DMACH2	RO	0x1	DMA Channel 2 When set, indicates μDMA channel 2 is available.
1	DMACH1	RO	0x1	DMA Channel 1 When set, indicates μDMA channel 1 is available.
0	DMACH0	RO	0x1	DMA Channel 0 When set, indicates μDMA channel 0 is available.

Register 114: Device Capabilities 8 (DC8), offset 0x02C

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

Important: This register is provided for legacy software support only.

The **ADC Peripheral Properties (ADCPP)** register should be used to determine how many input channels are available on the ADC module. A read of this register correctly identifies if legacy channels are present but software must use the **ADCPP** register to determine if a channel is present that is not supported by the **DCn** registers.

Device Capabilities 8 (DC8)

Base 0x400F.E000
Offset 0x02C
Type RO, reset 0x0FFF.0FFF

	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	0	0	0	0	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	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31	ADC1AIN15	RO	0x0	ADC Module 1 AIN15 Pin Present When set, indicates that ADC module 1 input pin 15 is present.
30	ADC1AIN14	RO	0x0	ADC Module 1 AIN14 Pin Present When set, indicates that ADC module 1 input pin 14 is present.
29	ADC1AIN13	RO	0x0	ADC Module 1 AIN13 Pin Present When set, indicates that ADC module 1 input pin 13 is present.
28	ADC1AIN12	RO	0x0	ADC Module 1 AIN12 Pin Present When set, indicates that ADC module 1 input pin 12 is present.
27	ADC1AIN11	RO	0x1	ADC Module 1 AIN11 Pin Present When set, indicates that ADC module 1 input pin 11 is present.
26	ADC1AIN10	RO	0x1	ADC Module 1 AIN10 Pin Present When set, indicates that ADC module 1 input pin 10 is present.
25	ADC1AIN9	RO	0x1	ADC Module 1 AIN9 Pin Present When set, indicates that ADC module 1 input pin 9 is present.
24	ADC1AIN8	RO	0x1	ADC Module 1 AIN8 Pin Present When set, indicates that ADC module 1 input pin 8 is present.
23	ADC1AIN7	RO	0x1	ADC Module 1 AIN7 Pin Present When set, indicates that ADC module 1 input pin 7 is present.
22	ADC1AIN6	RO	0x1	ADC Module 1 AIN6 Pin Present When set, indicates that ADC module 1 input pin 6 is present.

Bit/Field	Name	Type	Reset	Description
21	ADC1AIN5	RO	0x1	ADC Module 1 AIN5 Pin Present When set, indicates that ADC module 1 input pin 5 is present.
20	ADC1AIN4	RO	0x1	ADC Module 1 AIN4 Pin Present When set, indicates that ADC module 1 input pin 4 is present.
19	ADC1AIN3	RO	0x1	ADC Module 1 AIN3 Pin Present When set, indicates that ADC module 1 input pin 3 is present.
18	ADC1AIN2	RO	0x1	ADC Module 1 AIN2 Pin Present When set, indicates that ADC module 1 input pin 2 is present.
17	ADC1AIN1	RO	0x1	ADC Module 1 AIN1 Pin Present When set, indicates that ADC module 1 input pin 1 is present.
16	ADC1AIN0	RO	0x1	ADC Module 1 AIN0 Pin Present When set, indicates that ADC module 1 input pin 0 is present.
15	ADC0AIN15	RO	0x0	ADC Module 0 AIN15 Pin Present When set, indicates that ADC module 0 input pin 15 is present.
14	ADC0AIN14	RO	0x0	ADC Module 0 AIN14 Pin Present When set, indicates that ADC module 0 input pin 14 is present.
13	ADC0AIN13	RO	0x0	ADC Module 0 AIN13 Pin Present When set, indicates that ADC module 0 input pin 13 is present.
12	ADC0AIN12	RO	0x0	ADC Module 0 AIN12 Pin Present When set, indicates that ADC module 0 input pin 12 is present.
11	ADC0AIN11	RO	0x1	ADC Module 0 AIN11 Pin Present When set, indicates that ADC module 0 input pin 11 is present.
10	ADC0AIN10	RO	0x1	ADC Module 0 AIN10 Pin Present When set, indicates that ADC module 0 input pin 10 is present.
9	ADC0AIN9	RO	0x1	ADC Module 0 AIN9 Pin Present When set, indicates that ADC module 0 input pin 9 is present.
8	ADC0AIN8	RO	0x1	ADC Module 0 AIN8 Pin Present When set, indicates that ADC module 0 input pin 8 is present.
7	ADC0AIN7	RO	0x1	ADC Module 0 AIN7 Pin Present When set, indicates that ADC module 0 input pin 7 is present.
6	ADC0AIN6	RO	0x1	ADC Module 0 AIN6 Pin Present When set, indicates that ADC module 0 input pin 6 is present.
5	ADC0AIN5	RO	0x1	ADC Module 0 AIN5 Pin Present When set, indicates that ADC module 0 input pin 5 is present.
4	ADC0AIN4	RO	0x1	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	0x1	ADC Module 0 AIN3 Pin Present When set, indicates that ADC module 0 input pin 3 is present.
2	ADC0AIN2	RO	0x1	ADC Module 0 AIN2 Pin Present When set, indicates that ADC module 0 input pin 2 is present.
1	ADC0AIN1	RO	0x1	ADC Module 0 AIN1 Pin Present When set, indicates that ADC module 0 input pin 1 is present.
0	ADC0AIN0	RO	0x1	ADC Module 0 AIN0 Pin Present When set, indicates that ADC module 0 input pin 0 is present.

Register 115: 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.

Important: This register is provided for legacy software support only.

The peripheral-specific Software Reset registers (such as **SRWD**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this legacy register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as Watchdog 1), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Software Reset Control 0 (SRCR0)

Base 0x400F.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			WDT1	reserved			CAN0	reserved						ADC1	ADC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												WDT0	reserved		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:29	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	0x0	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.
27:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	CAN0	RO	0x0	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.

Bit/Field	Name	Type	Reset	Description
17	ADC1	RO	0x0	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	RO	0x0	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.
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT0	RO	0x0	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 116: 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.

Important: This register is provided for legacy software support only.

The peripheral-specific Software Reset registers (such as **SRTIMER**) should be used to reset specific peripherals. A write to this register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as TIMER0), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Note that the **Software Reset Analog Comparator (SRACMP)** register has only one bit to set the analog comparator module. Resetting the module resets all the blocks. If any of the `COMPn` bits are set, the entire analog comparator module is reset. It is not possible to reset the blocks individually.

Software Reset Control 1 (SRCR1)

Base 0x400F.E000
Offset 0x044
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of 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	COMP1	RO	0x0	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	RO	0x0	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.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
19	TIMER3	RO	0x0	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	RO	0x0	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	RO	0x0	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	RO	0x0	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	RO	0x0	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	RO	0x0	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	RO	0x0	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	RO	0x0	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.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2	UART2	RO	0x0	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	RO	0x0	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	RO	0x0	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 117: 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.

Important: This register is provided for legacy software support only.

The peripheral-specific Software Reset registers (such as **SRDMA**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as the μDMA), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Software Reset Control 2 (SRCR2)

Base 0x400F.E000
Offset 0x048
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	USB0	RO	0x0	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	RO	0x0	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.
12:7	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	GPIOG	RO	0x0	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	RO	0x0	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	RO	0x0	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	RO	0x0	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	RO	0x0	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	RO	0x0	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	RO	0x0	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.

Register 118: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100

This register controls the clock gating logic in normal Run mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes. Note that there must be a delay of 3 system clocks after a module clock is enabled before any registers in that module are accessed.

Important: This register is provided for legacy software support only.

The peripheral-specific Run Mode Clock Gating Control registers (such as **RCGCWD**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as Watchdog 1), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Likewise, the **ADC Peripheral Configuration (ADCPC)** register should be used to configure the ADC sample rate. However, to support legacy software, the MAXADCnSPD fields are available. A write to these legacy fields also writes the corresponding field in the peripheral-specific register. If a field is changed by writing to this register, it can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support rates that are not available in this register. If software uses a peripheral-specific register to set the ADC rate, the write causes proper operation, but the value of that field is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Run Mode Clock Gating Control Register 0 (RCGC0)

Base 0x400F.E000
Offset 0x100
Type RO, reset 0x0000.0040

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved			CAN0	reserved						ADC1	ADC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	reserved	reserved		WDT0	reserved		
Type	RO	RO	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
31:29	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	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	CAN0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
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	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	ADC0	RO	0x0	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 unclocked and disabled. If the module is unclocked, 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.

Bit/Field	Name	Type	Reset	Description
11:10	MAXADC1SPD	RO	0x0	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): Value Description 0x0 125K samples/second 0x1 250K samples/second 0x2 500K samples/second 0x3 1M samples/second
9:8	MAXADC0SPD	RO	0x0	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): Value Description 0x0 125K samples/second 0x1 250K samples/second 0x2 500K samples/second 0x3 1M 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	reserved	RO	0x1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
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	RO	0x0	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 119: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104

This register controls the clock gating logic in normal Run mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC1** is the clock configuration register for running operation, **SCGC1** for Sleep operation, and **DCGC1** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes. Note that there must be a delay of 3 system clocks after a module clock is enabled before any registers in that module are accessed.

Important: This register is provided for legacy software support only.

The peripheral-specific Run Mode Clock Gating Control registers (such as **RCGCTIMER**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as Timer 0), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Run Mode Clock Gating Control Register 1 (RCGC1)

Base 0x400F.E000
Offset 0x104
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
25	COMP1	RO	0x0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
24	COMP0	RO	0x0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 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.
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	0x0	Timer 3 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
18	TIMER2	RO	0x0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
17	TIMER1	RO	0x0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	TIMER0	RO	0x0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	RO	0x0	I2C1 Clock Gating Control This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
12	I2C0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	SSI0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 120: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108

This register controls the clock gating logic in normal Run mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC2** is the clock configuration register for running operation, **SCGC2** for Sleep operation, and **DCGC2** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes. Note that there must be a delay of 3 system clocks after a module clock is enabled before any registers in that module are accessed.

Important: This register is provided for legacy software support only.

The peripheral-specific Run Mode Clock Gating Control registers (such as **RCGCDMA**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as the µDMA), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Run Mode Clock Gating Control Register 2 (RCGC2)

Base 0x400F.E000
Offset 0x108
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
16	USB0	RO	0x0	USB0 Clock Gating Control This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:14	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	RO	0x0	Micro-DMA Clock Gating Control This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
12:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	GPIOG	RO	0x0	Port G Clock Gating Control This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
5	GPIOF	RO	0x0	Port F Clock Gating Control This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	GPIOE	RO	0x0	Port E Clock Gating Control Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	GPIOD	RO	0x0	Port D Clock Gating Control Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
2	GPIOC	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	GPIOB	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
0	GPIOA	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 121: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110

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

Important: This register is provided for legacy software support only.

The peripheral-specific Sleep Mode Clock Gating Control registers (such as **SCGCWD**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as Watchdog 1), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Sleep Mode Clock Gating Control Register 0 (SCGC0)

Base 0x400F.E000
Offset 0x110
Type RO, reset 0x0000.0040

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved			CAN0	reserved						ADC1	ADC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									reserved	reserved		WDT0	reserved		
Type	RO	RO	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
31:29	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	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
27:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	CAN0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
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	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	ADC0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	reserved	RO	0x1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
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	RO	0x0	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 122: 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.

Important: This register is provided for legacy software support only.

The peripheral-specific Sleep Mode Clock Gating Control registers (such as **SCGCTIMER**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as Timer 0), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Sleep Mode Clock Gating Control Register 1 (SCGC1)

Base 0x400F.E000
Offset 0x114
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of 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	COMP1	RO	0x0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
24	COMP0	RO	0x0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 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.
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	0x0	Timer 3 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
18	TIMER2	RO	0x0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
17	TIMER1	RO	0x0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	TIMER0	RO	0x0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	RO	0x0	I2C1 Clock Gating Control This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
5	SSI1	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	SSI0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 123: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118

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

Important: This register is provided for legacy software support only.

The peripheral-specific Sleep Mode Clock Gating Control registers (such as **SCGCDMA**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as the μDMA), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Sleep Mode Clock Gating Control Register 2 (SCGC2)

Base 0x400F.E000
Offset 0x118
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	USB0	RO	0x0	USB0 Clock Gating Control This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
15:14	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	RO	0x0	Micro-DMA Clock Gating Control This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
12:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	GPIOG	RO	0x0	Port G Clock Gating Control This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
5	GPIOF	RO	0x0	Port F Clock Gating Control This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	GPIOE	RO	0x0	Port E Clock Gating Control Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	GPIOD	RO	0x0	Port D Clock Gating Control Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
2	GPIOC	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	GPIOB	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	GPIOA	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 124: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120

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

Important: This register is provided for legacy software support only.

The peripheral-specific Deep Sleep Mode Clock Gating Control registers (such as **DCGCWD**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as Watchdog 1), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

Base 0x400F.E000
Offset 0x120
Type RO, reset 0x0000.0040

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved			CAN0	reserved						ADC1	ADC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									reserved	reserved		WDT0	reserved		
Type	RO	RO	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
31:29	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	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
27:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	CAN0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
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	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	ADC0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	reserved	RO	0x1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
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	RO	0x0	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 125: 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.

Important: This register is provided for legacy software support only.

The peripheral-specific Deep Sleep Mode Clock Gating Control registers (such as **DCGCTIMER**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as Timer 0), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

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

Base 0x400F.E000
Offset 0x124
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of 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	COMP1	RO	0x0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
24	COMP0	RO	0x0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 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.
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	0x0	Timer 3 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
18	TIMER2	RO	0x0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
17	TIMER1	RO	0x0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	TIMER0	RO	0x0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	RO	0x0	I2C1 Clock Gating Control This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
5	SSI1	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	SSI0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 126: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128

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

Important: This register is provided for legacy software support only.

The peripheral-specific Deep Sleep Mode Clock Gating Control registers (such as **DCGCDMA**) should be used to reset specific peripherals. A write to this legacy register also writes the corresponding bit in the peripheral-specific register. Any bits that are changed by writing to this register can be read back correctly with a read of this register. Software must use the peripheral-specific registers to support modules that are not present in the legacy registers. If software uses a peripheral-specific register to write a legacy peripheral (such as the µDMA), the write causes proper operation, but the value of that bit is not reflected in this register. If software uses both legacy and peripheral-specific register accesses, the peripheral-specific registers must be accessed by read-modify-write operations that affect only peripherals that are not present in the legacy registers. In this manner, both the peripheral-specific and legacy registers have coherent information.

Deep Sleep Mode Clock Gating Control Register 2 (DCGC2)

Base 0x400F.E000
Offset 0x128
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	USB0	RO	0x0	USB0 Clock Gating Control This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
15:14	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	RO	0x0	Micro-DMA Clock Gating Control This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
12:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	GPIOG	RO	0x0	Port G Clock Gating Control This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
5	GPIOF	RO	0x0	Port F Clock Gating Control This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	GPIOE	RO	0x0	Port E Clock Gating Control Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	GPIOD	RO	0x0	Port D Clock Gating Control Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
2	GPIOC	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	GPIOB	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	GPIOA	RO	0x0	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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 127: Device Capabilities 9 (DC9), offset 0x190

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

Important: This register is provided for legacy software support only.

The **ADC Peripheral Properties (ADCPP)** register should be used to determine how many digital comparators are available on the ADC module. A read of this register correctly identifies if legacy comparators are present. Software must use the **ADCPP** register to determine if a comparator that is not supported by the **DCn** registers is present.

Device Capabilities 9 (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	0x1	ADC1 DC7 Present When set, indicates that ADC module 1 Digital Comparator 7 is present.
22	ADC1DC6	RO	0x1	ADC1 DC6 Present When set, indicates that ADC module 1 Digital Comparator 6 is present.
21	ADC1DC5	RO	0x1	ADC1 DC5 Present When set, indicates that ADC module 1 Digital Comparator 5 is present.
20	ADC1DC4	RO	0x1	ADC1 DC4 Present When set, indicates that ADC module 1 Digital Comparator 4 is present.
19	ADC1DC3	RO	0x1	ADC1 DC3 Present When set, indicates that ADC module 1 Digital Comparator 3 is present.
18	ADC1DC2	RO	0x1	ADC1 DC2 Present When set, indicates that ADC module 1 Digital Comparator 2 is present.
17	ADC1DC1	RO	0x1	ADC1 DC1 Present When set, indicates that ADC module 1 Digital Comparator 1 is present.
16	ADC1DC0	RO	0x1	ADC1 DC0 Present When set, indicates that ADC module 1 Digital Comparator 0 is present.

Bit/Field	Name	Type	Reset	Description
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	0x1	ADC0 DC7 Present When set, indicates that ADC module 0 Digital Comparator 7 is present.
6	ADC0DC6	RO	0x1	ADC0 DC6 Present When set, indicates that ADC module 0 Digital Comparator 6 is present.
5	ADC0DC5	RO	0x1	ADC0 DC5 Present When set, indicates that ADC module 0 Digital Comparator 5 is present.
4	ADC0DC4	RO	0x1	ADC0 DC4 Present When set, indicates that ADC module 0 Digital Comparator 4 is present.
3	ADC0DC3	RO	0x1	ADC0 DC3 Present When set, indicates that ADC module 0 Digital Comparator 3 is present.
2	ADC0DC2	RO	0x1	ADC0 DC2 Present When set, indicates that ADC module 0 Digital Comparator 2 is present.
1	ADC0DC1	RO	0x1	ADC0 DC1 Present When set, indicates that ADC module 0 Digital Comparator 1 is present.
0	ADC0DC0	RO	0x1	ADC0 DC0 Present When set, indicates that ADC module 0 Digital Comparator 0 is present.

Register 128: Non-Volatile Memory Information (NVMSTAT), offset 0x1A0

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

Important: This register is provided for legacy software support only.

The **ROM Third-Party Software (ROMSWMAP)** register should be used to determine the presence of third-party software in the on-chip ROM on this microcontroller. A read of the `TPSW` bit in this register correctly identifies the presence of legacy third-party software. Software should use the **ROMSWMAP** register for software that is not on legacy devices.

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	0x1	32 Word Flash Write Buffer Available When set, indicates that the 32 word Flash memory write buffer feature is available.

6 System Exception Module

This module is an AHB peripheral that handles system-level Cortex-M4 FPU exceptions. For functions with registers mapped into this aperture, if the function is not available on a device, then all writes to the associated registers are ignored and reads return zeros.

6.1 Functional Description

The System Exception module provides control and status of the system-level interrupts. All the interrupt events are ORed together before being sent to the interrupt controller, so the System Exception module can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **System Exception Masked Interrupt Status (SYSEXCMIS)** register. The interrupt events that can trigger a controller-level interrupt are defined in the **System Exception Interrupt Mask (SYSEXCIM)** register by setting the corresponding interrupt mask bits. If interrupts are not used, the raw interrupt status is always visible via the **System Exception Raw Interrupt Status (SYSEXCRIS)** register. Interrupts are always cleared (for both the **SYSEXCMIS** and **SYSEXCRIS** registers) by writing a 1 to the corresponding bit in the **System Exception Interrupt Clear (SYSEXCIC)** register.

6.2 Register Map

Table 6-1 on page 449 lists the System Exception module registers. The offset listed is a hexadecimal increment to the register's address, relative to the System Exception base address of 0x400F.9000.

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

Table 6-1. System Exception Register Map

Offset	Name	Type	Reset	Description	See page
0x000	SYSEXCRIS	RO	0x0000.0000	System Exception Raw Interrupt Status	450
0x004	SYSEXCIM	RW	0x0000.0000	System Exception Interrupt Mask	452
0x008	SYSEXCMIS	RO	0x0000.0000	System Exception Masked Interrupt Status	454
0x00C	SYSEXCIC	W1C	0x0000.0000	System Exception Interrupt Clear	456

6.3 Register Descriptions

All addresses given are relative to the System Exception base address of 0x400F.9000.

Register 1: System Exception Raw Interrupt Status (SYSEXCRIS), offset 0x000

The **SYSEXCRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

System Exception Raw Interrupt Status (SYSEXCRIS)

Base 0x400F.9000
Offset 0x000
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										FPIXCRIS	FPOFCRIS	FPUFCRIS	FPIOCRIS	FPDZCRIS	FPIDCRIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	FPIXCRIS	RO	0	Floating-Point Inexact Exception Raw Interrupt Status Value Description 0 No interrupt 1 A floating-point inexact exception has occurred. This bit is cleared by writing a 1 to the `IXCIC` bit in the **SYSEXCIC** register.
4	FPOFCRIS	RO	0	Floating-Point Overflow Exception Raw Interrupt Status Value Description 0 No interrupt 1 A floating-point overflow exception has occurred. This bit is cleared by writing a 1 to the `OFCIC` bit in the **SYSEXCIC** register.
3	FPUFCRIS	RO	0	Floating-Point Underflow Exception Raw Interrupt Status Value Description 0 No interrupt 1 A floating-point underflow exception has occurred. This bit is cleared by writing a 1 to the `UFCIC` bit in the **SYSEXCIC** register.

Bit/Field	Name	Type	Reset	Description
2	FPIOCRIS	RO	0	Floating-Point Invalid Operation Raw Interrupt Status Value Description 0 No interrupt 1 A floating-point invalid operation exception has occurred. This bit is cleared by writing a 1 to the `IOCIC` bit in the **SYSEXCIC** register.
1	FPDZCRIS	RO	0	Floating-Point Divide By 0 Exception Raw Interrupt Status Value Description 0 No interrupt 1 A floating-point divide by 0 exception has occurred. This bit is cleared by writing a 1 to the `DZCIC` bit in the **SYSEXCIC** register.
0	FPIDCRIS	RO	0	Floating-Point Input Denormal Exception Raw Interrupt Status Value Description 0 No interrupt 1 A floating-point input denormal exception has occurred. This bit is cleared by writing a 1 to the `IDCIC` bit in the **SYSEXCIC** register.

Register 2: System Exception Interrupt Mask (SYSEXCIM), offset 0x004

The **SYSEXCIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Setting a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Clearing a bit prevents the raw interrupt signal from being sent to the interrupt controller.

System Exception Interrupt Mask (SYSEXCIM)

Base 0x400F.9000
Offset 0x004
Type RW, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										FPIXCIM	FPOFCIM	FPUFCIM	FPIOCIM	FPDZCIM	FPIDCIM
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RW	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	FPIXCIM	RW	0	Floating-Point Inexact Exception Interrupt Mask Value Description 0 The `FPIXCRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `FPISCRIS` bit in the **SYSEXCRIS** register is set.
4	FPOFCIM	RW	0	Floating-Point Overflow Exception Interrupt Mask Value Description 0 The `FPOFCIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `FPOFCRIS` bit in the **SYSEXCRIS** register is set.
3	FPUFCIM	RW	0	Floating-Point Underflow Exception Interrupt Mask Value Description 0 The `FPUFCRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `FPUFCRIS` bit in the **SYSEXCRIS** register is set.

Bit/Field	Name	Type	Reset	Description
2	FPIOCIM	RW	0	Floating-Point Invalid Operation Interrupt Mask Value Description 0 The `FPIOCRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `FPIOCRIS` bit in the **SYSEXCRIS** register is set.
1	FPDZCIM	RW	0	Floating-Point Divide By 0 Exception Interrupt Mask Value Description 0 The `FPDZCRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `FPDZCRIS` bit in the **SYSEXCRIS** register is set.
0	FPIDCIM	RW	0	Floating-Point Input Denormal Exception Interrupt Mask Value Description 0 The `FPIDCRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `FPIDCRIS` bit in the **SYSEXCRIS** register is set.

Register 3: System Exception Masked Interrupt Status (SYSEXCMIS), offset 0x008

The **SYSEXCMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

System Exception Masked Interrupt Status (SYSEXCMIS)

Base 0x400F.9000
Offset 0x008
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										FPIXCMIS	FPOFCMIS	FPUFCMIS	FPIOCMIS	FPDZCMIS	FPIDCMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	FPIXCMIS	RO	0	Floating-Point Inexact Exception Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to an inexact exception. This bit is cleared by writing a 1 to the `FPIXCIC` bit in the **SYSEXCIC** register.
4	FPOFCMIS	RO	0	Floating-Point Overflow Exception Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to an overflow exception. This bit is cleared by writing a 1 to the `FPOFCIC` bit in the **SYSEXCIC** register.
3	FPUFCMIS	RO	0	Floating-Point Underflow Exception Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to an underflow exception. This bit is cleared by writing a 1 to the `FPUFCIC` bit in the **SYSEXCIC** register.

Bit/Field	Name	Type	Reset	Description
2	FPIOCMIS	RO	0	Floating-Point Invalid Operation Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to an invalid operation. This bit is cleared by writing a 1 to the `FPIOCIC` bit in the **SYSEXCIC** register.
1	FPDZCMIS	RO	0	Floating-Point Divide By 0 Exception Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to a divide by 0 exception. This bit is cleared by writing a 1 to the `FPDZCIC` bit in the **SYSEXCIC** register.
0	FPIDCMIS	RO	0	Floating-Point Input Denormal Exception Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to an input denormal exception. This bit is cleared by writing a 1 to the `FPIDCIC` bit in the **SYSEXCIC** register.

Register 4: System Exception Interrupt Clear (SYSEXCIC), offset 0x00C

The **SYSEXCIC** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

System Exception Interrupt Clear (SYSEXCIC)

Base 0x400F.9000
Offset 0x00C
Type W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	W1C	W1C	W1C	W1C	W1C	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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										FPIXCIC	FPOFCIC	FPUFCIC	FPIOCIC	FPDZCIC	FPIDCIC
Type	W1C	W1C	W1C	W1C	W1C	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
31:6	reserved	W1C	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	FPIXCIC	W1C	0	Floating-Point Inexact Exception Interrupt Clear Writing a 1 to this bit clears the `FPIXCRIS` bit in the **SYSEXCRIS** register and the `FPIXCMIS` bit in the **SYSEXCMIS** register.
4	FPOFCIC	W1C	0	Floating-Point Overflow Exception Interrupt Clear Writing a 1 to this bit clears the `FPOFCRIS` bit in the **SYSEXCRIS** register and the `FPOFCMIS` bit in the **SYSEXCMIS** register.
3	FPUFCIC	W1C	0	Floating-Point Underflow Exception Interrupt Clear Writing a 1 to this bit clears the `FPUFCRIS` bit in the **SYSEXCRIS** register and the `FPUFCMIS` bit in the **SYSEXCMIS** register.
2	FPIOCIC	W1C	0	Floating-Point Invalid Operation Interrupt Clear Writing a 1 to this bit clears the `FPIOCRIS` bit in the **SYSEXCRIS** register and the `FPIOCMIS` bit in the **SYSEXCMIS** register.
1	FPDZCIC	W1C	0	Floating-Point Divide By 0 Exception Interrupt Clear Writing a 1 to this bit clears the `FPDZCRIS` bit in the **SYSEXCRIS** register and the `FPDZCMIS` bit in the **SYSEXCMIS** register.
0	FPIDCIC	W1C	0	Floating-Point Input Denormal Exception Interrupt Clear Writing a 1 to this bit clears the `FPIDCRIS` bit in the **SYSEXCRIS** register and the `FPIDCMIS` bit in the **SYSEXCMIS** register.

7 Internal Memory

The TM4C1232C3PM microcontroller comes with 12 KB of bit-banded SRAM, internal ROM, 32 KB of Flash memory, and 2KB of EEPROM. The Flash memory controller provides a user-friendly interface, making Flash memory programming a simple task. Flash memory is organized in 1-KB independently erasable blocks and memory protection can be applied to the Flash memory on a 2-KB block basis. The EEPROM module provides a well-defined register interface to support accesses to the EEPROM with both a random access style of read and write as well as a rolling or sequential access scheme. A password model allows the application to lock one or more EEPROM blocks to control access on 16-word boundaries.

7.1 Block Diagram

Figure 7-1 on page 457 illustrates the internal SRAM, ROM, and Flash memory blocks and control logic. The dashed boxes in the figure indicate registers residing in the System Control module.

Figure 7-1. Internal Memory Block Diagram

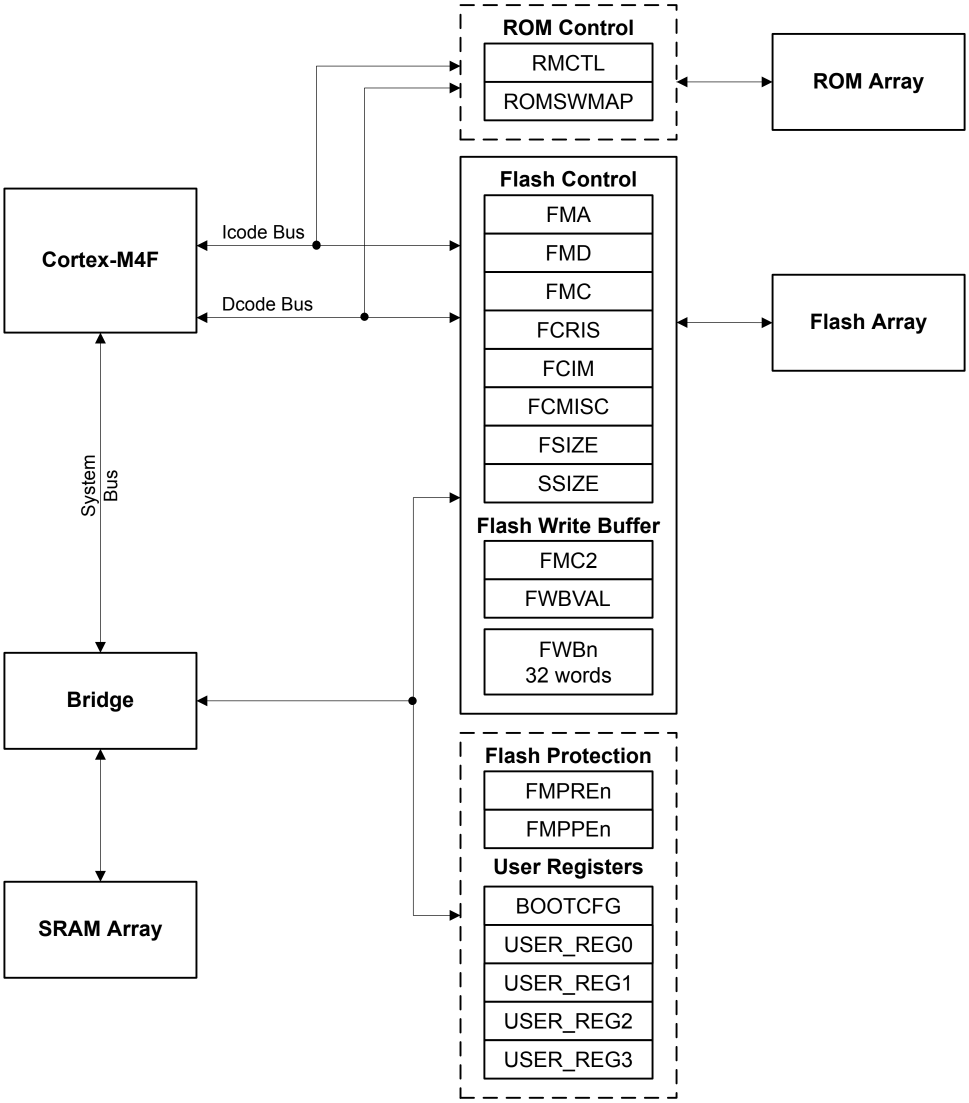
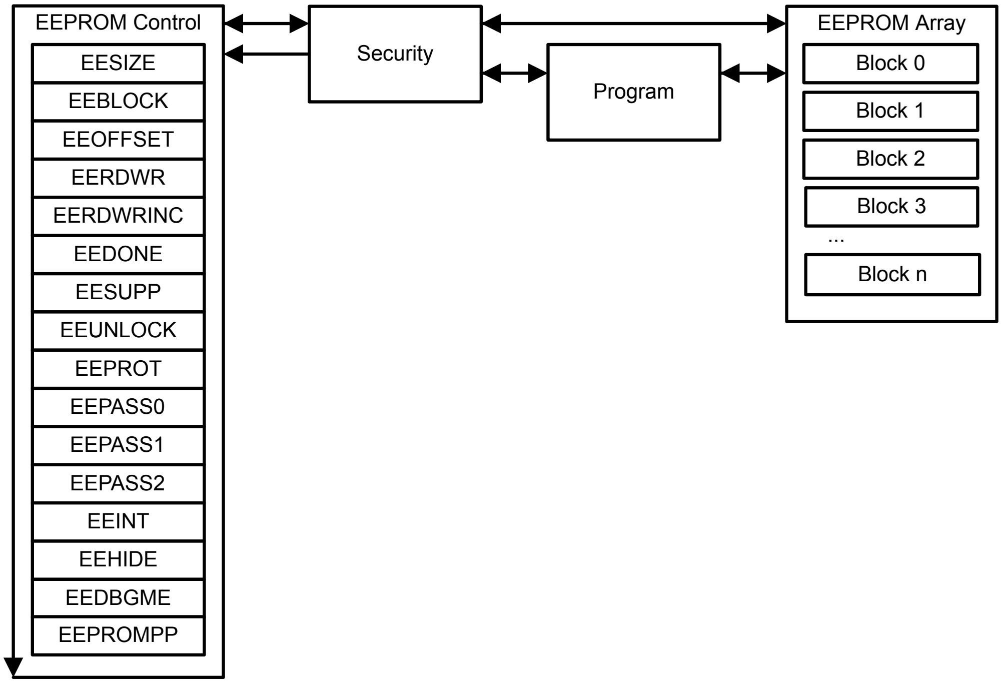


Figure 7-2 on page 458 illustrates the internal EEPROM block and control logic. The EEPROM block is connected to the AHB bus.

Figure 7-2. EEPROM Block Diagram



7.2 Functional Description

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

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

7.2.1 SRAM

The internal SRAM of the TM4C1232C3PM device is located at address 0x2000.0000 of the device memory map. To reduce the number of time consuming read-modify-write (RMW) operations, ARM provides bit-banding technology in the processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation. The bit-band base is located at address 0x2200.0000.

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

```
bit-band alias = bit-band base + (byte offset * 32) + (bit number * 4)
```

For example, if bit 3 at address 0x2000.1000 is to be modified, the bit-band alias is calculated as:

```
0x2200.0000 + (0x1000 * 32) + (3 * 4) = 0x2202.000C
```

With the alias address calculated, an instruction performing a read/write to address 0x2202.000C allows direct access to only bit 3 of the byte at address 0x2000.1000.

For details about bit-banding, see “Bit-Banding” on page 86.

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

7.2.2 ROM

The internal ROM of the TM4C1232C3PM device is located at address 0x0100.0000 of the device memory map. Detailed information on the ROM contents can be found in the *Tiva™ C Series TM4C123x ROM User's Guide (literature number SPMU367)*.

The ROM contains the following components:

- TivaWare™ Boot Loader and vector table
- TivaWare Peripheral Driver Library (DriverLib) release for product-specific peripherals and interfaces
- Advanced Encryption Standard (AES) cryptography tables
- Cyclic Redundancy Check (CRC) error detection functionality

The boot loader is used as an initial program loader (when the Flash memory is empty) as well as an application-initiated firmware upgrade mechanism (by calling back to the boot loader). The Peripheral Driver Library APIs in ROM can be called by applications, reducing Flash memory requirements and freeing the Flash memory to be used for other purposes (such as additional features in the application). Advance Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government and Cyclic Redundancy Check (CRC) is a technique to validate if a block of data has the same contents as when previously checked.

7.2.2.1 Boot Loader Overview

The TivaWare Boot Loader is used to download code to the Flash memory of a device without the use of a debug interface. When the core is reset, the user has the opportunity to direct the core to execute the ROM Boot Loader or the application in Flash memory by using any GPIO signal in Ports A-H as configured in the **Boot Configuration (BOOTCFG)** register (see page 513).

At reset, the following sequence is performed:

1. The **BOOTCFG** register is read. If the EN bit is clear, the ROM Boot Loader is executed.
2. In the ROM Boot Loader, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
3. If the EN bit is set or the status doesn't match the specified polarity, the data at address 0x0000.0004 is read, and if the data at this address is 0xFFFF.FFFF, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
4. If there is data at address 0x0000.0004 that is not 0xFFFF.FFFF, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

The boot loader uses a simple packet interface to provide synchronous communication with the device. The speed of the boot loader is determined by the internal oscillator (PIOSC) frequency as it does not enable the PLL. The following serial interfaces can be used:

- UART0
- SSI0
- I^{2}C0
- USB

The data format and communication protocol are identical for the UART0, SSI0, and I2C0 interfaces.

Note: The Flash-memory-resident version of the boot loader also supports CAN.

See the *TivaWare™ Boot Loader for C Series User's Guide (literature number SPMU301)* for information on the boot loader software. The USB boot loader uses the standard Device Firmware Upgrade USB device class.

Considerations When Using the UART Boot Loader in ROM

`U0Tx` is not driven by the ROM boot loader until the auto-bauding process has completed. If `U0Tx` is floating during this time, the receiver it is connected to may see transitions on the signal, which could be interpreted by its UART as valid characters. To handle this situation, put a pull-up or pull-down on `U0Tx`, providing a defined state for the signal until the ROM boot loader begins driving `U0Tx`. A pull-up is preferred as it indicates that the UART is idle, rather than a pull-down, which indicates a break condition.

7.2.2.2 TivaWare Peripheral Driver Library

The TivaWare Peripheral Driver Library contains a file called `driverlib/rom.h` that assists with calling the peripheral driver library functions in the ROM. The detailed description of each function is available in the *Tiva™ C Series TM4C123x ROM User’s Guide (literature number SPMU367)*. See the "Using the ROM" chapter of the *TivaWare™ Peripheral Driver Library for C Series User's Guide (literature number SPMU298)* for more details on calling the ROM functions and using `driverlib/rom.h`. The `driverlib/rom_map.h` header file is also provided to aid portability when using different Tiva™ C Series devices which might have a different subset of DriverLib functions in ROM. The `driverlib/rom_map.h` header file uses build-time labels to route function calls to the ROM if those functions are available on a given device, otherwise, it routes to Flash-resident versions of the functions.

A table at the beginning of the ROM points to the entry points for the APIs that are provided in the ROM. Accessing the API through these tables provides scalability; while the API locations may change in future versions of the ROM, the API tables will not. The tables are split into two levels; the main table contains one pointer per peripheral which points to a secondary table that contains one pointer per API that is associated with that peripheral. The main table is located at 0x0100.0010, right after the Cortex-M4F vector table in the ROM.

DriverLib functions are described in detail in the *TivaWare™ Peripheral Driver Library for C Series User's Guide (literature number SPMU298)*.

Additional APIs are available for graphics and USB functions, but are not preloaded into ROM. The TivaWare Graphics Library provides a set of graphics primitives and a widget set for creating graphical user interfaces on Tiva™ C Series microcontroller-based boards that have a graphical display (for more information, see the *TivaWare™ Graphics Library for C Series User's Guide (literature number SPMU300)*). The TivaWare USB Library is a set of data types and functions for creating USB Device, Host or On-The-Go (OTG) applications on Tiva™ C Series microcontroller-based boards (for more information, see the *TivaWare™ USB Library for C Series User's Guide (literature number SPMU297)*).

7.2.2.3 Advanced Encryption Standard (AES) Cryptography Tables

AES is a strong encryption method with reasonable performance and size. AES is fast in both hardware and software, is fairly easy to implement, and requires little memory. AES is ideal for applications that can use prearranged keys, such as setup during manufacturing or configuration. Four data tables used by the XySSL AES implementation are provided in the ROM. The first is the forward S-box substitution table, the second is the reverse S-box substitution table, the third is the forward polynomial table, and the final is the reverse polynomial table. See the *Tiva™ C Series TM4C123x ROM User's Guide (literature number SPMU367)* for more information on AES.

7.2.2.4 Cyclic Redundancy Check (CRC) Error Detection

The CRC technique can be used to validate correct receipt of messages (nothing lost or modified in transit), to validate data after decompression, to validate that Flash memory contents have not been changed, and for other cases where the data needs to be validated. A CRC is preferred over a simple checksum (for example, XOR all bits) because it catches changes more readily. See the *Tiva™ C Series TM4C123x ROM User's Guide (literature number SPMU367)* for more information on CRC.

7.2.3 Flash Memory

At system clock speeds of 40 MHz and below, the Flash memory is read in a single cycle. The Flash memory is organized as a set of 1-KB blocks that can be individually erased. An individual 32-bit word can be programmed to change bits from 1 to 0. In addition, a write buffer provides the ability to program 32 continuous words in Flash memory in half the time of programming the words individually. 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 a debugger.

7.2.3.1 Prefetch Buffer

The Flash memory controller has a prefetch buffer that is automatically used when the CPU frequency is greater than 40 MHz. In this mode, the Flash memory operates at half of the system clock. The prefetch buffer fetches two 32-bit words per clock allowing instructions to be fetched with no wait states while code is executing linearly. The fetch buffer includes a branch speculation mechanism that recognizes a branch and avoids extra wait states by not reading the next word pair. Also, short loop branches often stay in the buffer. As a result, some branches can be executed with no wait states. Other branches incur a single wait state.

7.2.3.2 Flash Memory Protection

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

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

The policies may be combined as shown in Table 7-1 on page 462.

Table 7-1. Flash Memory Protection Policy Combinations

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

A Flash memory access that attempts to read a read-protected block (**FMPREn** bit is set) is prohibited and generates a bus fault. A Flash memory access that attempts to program or erase a program-protected block (**FMPPEn** bit is set) is prohibited and can optionally generate an interrupt (by setting the AMASK bit in the **Flash Controller Interrupt Mask (FCIM)** register) to alert software developers of poorly behaving software during the development and debug phases.

The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. These settings create a policy of open access and programmability. The register bits may be changed by clearing the specific register bit. The changes are effective immediately, but 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 “Non-Volatile Register Programming” on page 465.

7.2.3.3 Execute-Only Protection

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

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

1. Use a compiler that allows literal data to be collected into a separate section that is put into one or more read-enabled flash blocks. Note that the LDR instruction may use a PC-relative address–-in which case the literal pool cannot be located outside the span of the offset–-or the software may reserve a register to point to the base address of the literal pool and the LDR offset is relative to the beginning of the pool.
2. Use a compiler that generates literal data from arithmetic instruction immediate data and subsequent computation.
3. Use method 1 or 2, but in assembly language, if the compiler does not support either method.

7.2.3.4 Read-Only Protection

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

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

7.2.3.5 Permanently Disabling Debug

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

The DBG0 and DBG1 bits of the **Boot Configuration (BOOTCFG)** register control whether the debug interface is turned on or off.

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

7.2.3.6 Interrupts

The Flash memory controller can generate interrupts when the following conditions are observed:

- Programming Interrupt - signals when a program or erase action is complete.
- Access Interrupt - signals when a program or erase action has been attempted on a 2-kB block of memory that is protected by its corresponding **FMPPEn** bit.

The interrupt events that can trigger a controller-level interrupt are defined in the **Flash Controller Masked Interrupt Status (FCMIS)** register (see page 482) by setting the corresponding MASK bits. If interrupts are not used, the raw interrupt status is always visible via the **Flash Controller Raw Interrupt Status (FCRIS)** register (see page 479).

Interrupts are always cleared (for both the **FCMIS** and **FCRIS** registers) by writing a 1 to the corresponding bit in the **Flash Controller Masked Interrupt Status and Clear (FCMISC)** register (see page 484).

7.2.3.7 Flash Memory Programming

The Tiva™ C Series devices provide a user-friendly interface for Flash memory programming. All erase/program operations are handled via three registers: **Flash Memory Address (FMA)**, **Flash Memory Data (FMD)**, and **Flash Memory Control (FMC)**. Note that if the debug capabilities of the microcontroller have been deactivated, resulting in a "locked" state, a recovery sequence must be performed in order to reactivate the debug module. See “Recovering a "Locked" Microcontroller” on page 194.

During a Flash memory operation (write, page erase, or mass erase) access to the Flash memory is inhibited. As a result, instruction and literal fetches are held off until the Flash memory operation is complete. If instruction execution is required during a Flash memory operation, the code that is executing must be placed in SRAM and executed from there while the flash operation is in progress.

Note: When programming Flash memory, the following characteristics of the memory must be considered:

- Only an erase can change bits from 0 to 1.
- A write can only change bits from 1 to 0. If the write attempts to change a 0 to a 1, the write fails and no bits are changed.
- A flash operation can be started before entering the Sleep or Deep-Sleep mode (using the wait for interrupt instruction, `WFI`). It can also be completed while in Sleep or Deep-Sleep. If the Flash program/erase event comes in succession to EEPROM access, the Flash event gets completed after waking from Sleep/Deep-Sleep and is started after the wake-up.

7.2.3.8 Basic Program / Erase Operations

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 to the **FMC** register. Depending on the value of the `KEY` bit in the **BOOTCFG** register, the value 0xA442 or 0x71D5 must be written into the `WRKEY` field for a Flash memory write to occur.
4. Poll the **FMC** register until the `WRITE` bit is cleared.

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 to the **FMC** register. Depending on the value of the `KEY` bit in the **BOOTCFG** register, the value 0xA442 or 0x71D5 must be written into the `WRKEY` field for a Flash memory write to occur.
3. Poll the **FMC** register until the `ERASE` bit is cleared or, alternatively, enable the programming interrupt using the `PMASK` bit in the **FCIM** register.

To perform a mass erase of the Flash memory

1. Write the Flash memory write key and the MERASE bit to the **FMC** register. Depending on the value of the KEY bit in the **BOOTCFG** register, the value 0xA442 or 0x71D5 must be written into the WRKEY field for a Flash memory write to occur.

2. Poll the **FMC** register until the MERASE bit is cleared or, alternatively, enable the programming interrupt using the PMASK bit in the **FCIM** register.

7.2.3.9 32-Word Flash Memory Write Buffer

A 32-word write buffer provides the capability to perform faster write accesses to the Flash memory by programming 2 32-bit words at a time, allowing 32 words to be programmed in the same time as 16 would take using the method described above. The data for the buffered write is written to the **Flash Write Buffer (FWBn)** registers.

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

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

1. Write the source data to the **FWBn** registers.

2. Write the target address to the **FMA** register. This must be a 32-word aligned address (that is, bits [6:0] in **FMA** must be 0s).

3. Write the Flash memory write key and the WRBUF bit to the **FMC2** register. Depending on the value of the KEY bit in the **BOOTCFG** register, the value 0xA442 or 0x71D5 must be written into the WRKEY field for a Flash memory write to occur.

4. Poll the **FMC2** register until the WRBUF bit is cleared or wait for the PMIS interrupt to be signaled.

7.2.3.10 Non-Volatile Register Programming

Note: The **Boot Configuration (BOOTCFG)** register requires a POR before the committed changes take effect.

This section discusses how to update the registers shown in Table 7-2 on page 466 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. With the exception of the **Boot Configuration (BOOTCFG)** register, the settings in these registers can be written, their functions verified, and their values read back before they are committed, at which point they become non-volatile. If a value in one of these registers has not been committed, a power-on reset restores the last committed value or the default value if the register has never been committed. Other types of reset have no effect. 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 194.

To write to a non-volatile register:

- Bits can only be changed from 1 to 0.

- For all registers except the **BOOTCFG** register, write the data to the register address provided in the register description. For the **BOOTCFG** register, write the data to the **FMD** register.
- The registers can be read to verify their contents. To verify what is to be stored in the **BOOTCFG** register, read the **FMD** register. Reading the **BOOTCFG** register returns the previously committed value or the default value if the register has never been committed.
- The new values are effectively immediately for all registers except **BOOTCFG**, as the new value for the register is not stored in the register until it has been committed.
- Prior to committing the register value, a power-on reset restores the last committed value or the default value if the register has never been committed.

To commit a new value to a non-volatile register:

- Write the data as described above.
- Write to the **FMA** register the value shown in Table 7-2 on page 466.
- Write the Flash memory write key and set the `COMT` bit in the **FMC** register. These values must be written to the **FMC** register at the same time.
- Committing a non-volatile register has the same timing as a write to regular Flash memory, defined by T_{PROG64}, as shown in Table 21-24 on page 1146. Software can poll the `COMT` bit in the **FMC** register to determine when the operation is complete, or an interrupt can be enabled by setting the `PMASK` bit in the **FCIM** register.
- When committing the **BOOTCFG** register, the `INVDRIS` bit in the **FCRIS** register is set if a bit that has already been committed as a 0 is attempted to be committed as a 1.
- Once the value has been committed, a power-on reset has no effect on the register contents.
- Changes to the **BOOTCFG** register are effective after the next power-on reset.
- Once the `NW` bit has been changed to 0 and committed, further changes to the **BOOTCFG** register are not allowed.

Important: After being committed, these registers can only be restored to their factory default values by performing the sequence described in “Recovering a "Locked" Microcontroller” on page 194. The mass erase of the main Flash memory array caused by the sequence is performed prior to restoring these registers.

Table 7-2. User-Programmable Flash Memory Resident Registers

Register to be Committed	FMA Value	Data Source
FMPRE0	0x0000.0000	FMPRE0
FMPPE0	0x0000.0001	FMPPE0
USER_REG0	0x8000.0000	USER_REG0
USER_REG1	0x8000.0001	USER_REG1
USER_REG2	0x8000.0002	USER_REG2
USER_REG3	0x8000.0003	USER_REG3
BOOTCFG	0x7510.0000	FMD

7.2.4 EEPROM

The TM4C1232C3PM microcontroller includes an EEPROM with the following features:

- 2Kbytes of memory accessible as 512 32-bit words
- 32 blocks of 16 words (64 bytes) each
- Built-in wear leveling
- Access protection per block
- Lock protection option for the whole peripheral as well as per block using 32-bit to 96-bit unlock codes (application selectable)
- Interrupt support for write completion to avoid polling
- Endurance of 500K writes (when writing at fixed offset in every alternate page in circular fashion) to 15M operations (when cycling through two pages) per each 2-page block.

7.2.4.1 Functional Description

The EEPROM module provides a well-defined register interface to support accesses to the EEPROM with both a random access style of read and write as well as a rolling or sequential access scheme.

A protection mechanism allows locking EEPROM blocks to prevent writes under a set of circumstances as well as reads under the same or different circumstances. The password model allows the application to lock one or more EEPROM blocks to control access on 16-word boundaries.

Important: The configuration of the system clock must not be changed while an EEPROM operation is in process. Software must wait until the `WORKING` bit in the **EEPROM Done Status (EEDONE)** register is clear before making any changes to the system clock.

Blocks

There are 32 blocks of 16 words each in the EEPROM. Bytes and half-words can be read, and these accesses do not have to occur on a word boundary. The entire word is read and any unneeded data is simply ignored. They are writable only on a word basis. To write a byte, it is necessary to read the word value, modify the appropriate byte, and write the word back.

Each block is addressable as an offset within the EEPROM, using a block select register. Each word is offset addressable within the selected block.

The current block is selected by the **EEPROM Current Block (EEBLOCK)** register. The current offset is selected and checked for validity by the **EEPROM Current Offset (EEOFFSET)** register. The application may write the **EEOFFSET** register any time, and it is also automatically incremented when the **EEPROM Read-Write with Increment (EERDWRINC)** register is accessed. However, the **EERDWRINC** register does not increment the block number, but instead wraps within the block.

Blocks are individually protectable. Attempts to read from a block for which the application does not have permission return 0xFFFF.FFFF. Attempts to write into a block for which the application does not have permission results in an error in the **EEDONE** register.

Timing Considerations

After enabling or resetting the EEPROM module, software must wait until the `WORKING` bit in the **EEDONE** register is clear before accessing any EEPROM registers.

In the event that there are Flash memory writes or erases and EEPROM writes active, it is possible for the EEPROM process to be interrupted by the Flash memory write/erase and then continue after the Flash memory write is completed. This action may change the amount of time that the EEPROM operation takes.

EEPROM operations must be completed before entering Sleep or Deep-Sleep mode. Ensure the EEPROM operations have completed by checking the **EEPROM Done Status (EEDONE)** register before issuing a `WFI` instruction to enter Sleep or Deep-Sleep.

Reads of words within a block are at direct speed, which means that wait states are automatically generated if the system clock is faster than the speed of the EEPROM. The read access time is specified in Table 21-25 on page 1146.

Writing the **EEOFFSET** register also does not incur any penalties.

Writing the **EEBLOCK** register is not delayed, but any attempt to access data within that block is delayed by 4 clocks after writing **EEBLOCK**. This time is used to load block specific information.

Writes to words within a block are delayed by a variable amount of time. The application may use an interrupt to be notified when the write is done, or alternatively poll for the done status in the **EEDONE** register. The variability ranges from the write timing of the EEPROM to the erase timing of EEPROM, where the erase timing is less than the write timing of most external EEPROMs.

Locking and Passwords

The EEPROM can be locked at both the module level and the block level. The lock is controlled by a password that is stored in the **EEPROM Password (EEPASSn)** registers and can be any 32-bit to 96-bit value other than all 1s. Block 0 is the master block, the password for block 0 protects the control registers as well as all other blocks. Each block can be further protected with a password for that block.

If a password is registered for block 0, then the whole module is locked at reset. The locking behavior is such that blocks 1 to 31 are inaccessible until block 0 is unlocked, and block 0 follows the rules defined by its protection bits. As a result, the **EEBLOCK** register cannot be changed from 0 until block 0 is unlocked.

A password registered with any block, including block 0, allows for protection rules that control access of that block based on whether it is locked or unlocked. Generally, the lock can be used to prevent write accesses when locked or can prevent read and write accesses when locked.

All password-protected blocks are locked at reset. To unlock a block, the correct password value must be written to the **EEPROM Unlock (EEUNLOCK)** register by writing to it one to three times to form the 32-bit, 64-bit, or 96-bit password registered using the **EEPASSn** register. The value used to configure the **EEPASS0** register must always be written last. For example, for a 96-bit password, the value used to configure the **EEPASS2** register must be written first, followed by the **EEPASS1** and the **EEPASS0** register values. A block or the module may be re-locked by writing 0xFFFF.FFFF to the **EEUNLOCK** register because 0xFFFF.FFFF is not a valid password.

Protection and Access Control

The protection bits provide discrete control of read and write access for each block which allows various protection models per block, including:

- Without password: Readable and writable at any time. This mode is the default when there is no password.

- Without password: Readable but not writable.

- With password: Readable, but only writable when unlocked by the password. This mode is the default when there is a password.
- With password: Readable or writable only when unlocked.
- With password: Readable only when unlocked, not writable.

Additionally, access protection may be applied based on the processor mode. This configuration allows for supervisor-only access or supervisor and user access, which is the default. Supervisor-only access mode also prevents access by the μDMA and Debugger.

Additionally, the master block may be used to control access protection for the protection mechanism itself. If access control for block 0 is for supervisor only, then the whole module may only be accessed in supervisor mode. In addition, the protection level for block 0 sets the minimum protection level for the entire EEPROM. For example, if the PROT field in the **EEPROT** register is configured to 0x1 for block 0, then block 1 could be configured with the PROT field to be 0x1, 0x2, or 0x3, but not 0x0.

Note that for blocks 1 to 31, they are inaccessible for read or write if block 0 has a password and it is not unlocked. If block 0 has a master password, then the strictest protection defined for block 0 or an individual block is implemented on the remaining blocks.

Hidden Blocks

Hiding provides a temporary form of protection. Every block except block 0 can be hidden, which prevents all accesses until the next reset.

This mechanism can allow a boot or initialization routine to access some data which is then made inaccessible to all further accesses. Because boot and initialization routines control the capabilities of the application, hidden blocks provide a powerful isolation of the data when debug is disabled.

A typical use model would be to have the initialization code store passwords, keys, and/or hashes to use for verification of the rest of the application. Once performed, the block is then hidden and made inaccessible until the next reset which then re-enters the initialization code.

Power and Reset Safety

Once the **EEDONE** register indicates that a location has been successfully written, the data is retained until that location is written again. There is no power or reset race after the **EEDONE** register indicates a write has completed.

Interrupt Control

The EEPROM module allows for an interrupt when a write completes to eliminate the need for polling. The interrupt can be used to drive an application ISR which can then write more words or verify completion. The interrupt mechanism is used any time the **EEDONE** register goes from working to done, whether because of an error or the successful completion of a program or erase operation. This interrupt mechanism works for data writes, writes to password and protection registers, forced erase by the **EEPROM Support Control and Status (EESUPP)** register, and mass erase using the **EEPROM Debug Mass Erase (EEDGBME)** register. The EEPROM interrupt is signaled to the core using the Flash memory interrupt vector. Software can determine that the source of the interrupt was the EEPROM by examining bit 2 of the **Flash Controller Masked Interrupt Status and Clear (FCMISC)** register.

Theory of Operation

The EEPROM operates using a traditional Flash bank model which implements EEPROM-type cells, but uses sector erase. Additionally, words are replicated in the pages to allow 500K+ erase cycles when needed, which means that each word has a latest version. As a result, a write creates a new version of the word in a new location, making the previous value obsolete.

Each sector contains two blocks. Each block contains locations for the active copy plus six redundant copies. Passwords, protection bits, and control data are all stored in the pages.

When a page runs out of room to store the latest version of a word, a copy buffer is used. The copy buffer copies the latest words of each block. The original page is then erased. Finally, the copy buffer contents are copied back to the page. This mechanism ensures that data cannot be lost due to power down, even during an operation. The EEPROM mechanism properly tracks all state information to provide complete safety and protection. Although it should not normally be possible, errors during programming can occur in certain circumstances, for example, the voltage rail dropping during programming. In these cases, the **EESUPP** register can be used to finish an operation as described in the section called “Error During Programming” on page 470.

Manual Copy Buffer Erase

The copy buffer is only used when a main block is full because a word has been written seven times and there is no more room to store its latest version. In this situation, the latest versions of all the words in the block are copied to the copy buffer, allowing the main block to be erased safely, providing power down safety. If the copy buffer itself is full, then it must first be erased, which adds extra time. By performing a manual erase of the copy buffer, this overhead does not occur during a future write access. The EREQ bit in the **EESUPP** register is set if the copy buffer must be erased. If so, the START bit can be written by the application to force the erase at a more convenient time. The **EEDONE** and **EEINT** registers can be used to detect completion.

Debug Mass Erase

The EEPROM debug mass erase allows the developer to mass erase the EEPROM. For the mass erase to occur correctly, there can be no active EEPROM operations. After the last EEPROM operation, the application must ensure that no EEPROM registers are updated, including modifying the **EEBLOCK** and the **EEOFFSET** registers without doing an actual read or write operation. To hold off these operations, the application should reset the EEPROM module by setting the R0 bit in the **EEPROM Software Reset (SREEPROM**) register, wait until WORKING bit in the **EEPROM Done Status (EEDONE)** register is clear, and then enable the debug mass erase by setting the ME bit in the **EEPROM Debug Mass Erase (EEDBGME)** register.

Error During Programming

Operations such as data-write, password set, protection set, and copy buffer erase may perform multiple operations. For example, a normal write performs two underlying writes: the control word write and the data write. If the control word writes but the data fails (for example, due to a voltage drop), the overall write fails with indication provided in the **EEDONE** register. Failure and the corrective action is broken down by the type of operation:

- If a normal write fails such that the control word is written but the data fails to write, the safe course of action is to retry the operation once the system is otherwise stable, for example, when the voltage is stabilized. After the retry, the control word and write data are advanced to the next location.

- If a password or protection write fails, the safe course of action is to retry the operation once the system is otherwise stable. In the event that multi-word passwords may be written outside of a manufacturing or bring-up mode, care must be taken to ensure all words are written in immediate succession. If not, then partial password unlock would need to be supported to recover.

- If the word write requires the block to be written to the copy buffer, then it is possible to fail or lose power during the subsequent operations. A control word mechanism is used to track what step the EEPROM was in if a failure occurs. If not completed, the **EESUPP** register indicates the partial completion, and the **EESUPP** START bit can be written to allow it to continue to completion.

- If a copy buffer erase fails or power is lost while erasing, the **EESUPP** register indicates it is not complete and allows it to be restarted

After a reset and prior to writing any data to the EEPROM, software must read the **EESUPP** register and check for the presence of any error condition which may indicate that a write or erase was in progress when the system was reset due to a voltage drop. If either the PRETRY or ERETRY bits are set, the peripheral should be reset by setting and then clearing the R0 bit in the **EEPROM Software Reset (SREEPROM)** register and waiting for the WORKING bit in the **EEDONE** register to clear before again checking the **EESUPP** register for error indicators. This procedure should allow the EEPROM to recover from the write or erase error. In very isolated cases, the **EESUPP** register may continue to register an error after this operation, in which case the reset should be repeated. After recovery, the application should rewrite the data which was being programmed when the initial failure occurred.

Soft Reset Handling

The following soft resets should not be asserted during an EEPROM program or erase operation:

- Software reset (SYSRESREQ)

- Software peripheral reset

- Watchdog reset

- MOSC failure reset

The WORKING bit of the **EEDONE** register can be checked before the reset is asserted to see if an EEPROM program or erase operation is occurring. Soft resets may occur when using a debugger and should be avoided during an EEPROM operation. A reset such as the Watchdog reset can be mapped to an external reset using a GPIO, or Hibernate can be entered, if time is not a concern.

Endurance

Endurance is per meta-block which is 2 blocks. Endurance is measured in two ways:

1. To the application, it is the number of writes that can be performed.

2. To the microcontroller, it is the number of erases that can be performed on the meta-block.

Because of the second measure, the number of writes depends on how the writes are performed. For example:

- One word can be written more than 500K times, but, these writes impact the meta-block that the word is within. As a result, writing one word 500K times, then trying to write a nearby word 500K times is not assured to work. To ensure success, the words should be written more in parallel.

- All words can be written in a sweep with a total of more than 500K sweeps which updates all words more than 500K times.

- Different words can be written such that any or all words can be written more than 500K times when write counts per word stay about the same. For example, offset 0 could be written 3 times, then offset 1 could be written 2 times, then offset 2 is written 4 times, then offset 1 is written twice, then offset 0 is written again. As a result, all 3 offsets would have 4 writes at the end of the sequence. This kind of balancing within 7 writes maximizes the endurance of different words within the same meta-block.

7.2.4.2 EEPROM Initialization and Configuration

Before writing to any EEPROM registers, the clock to the EEPROM module must be enabled through the **EEPROM Run Mode Clock Gating Control (RCGCEEPROM)** register (see page 332) and the following initialization steps must be executed:

1. Insert delay (6 cycles plus function call overhead).
2. Poll the `WORKING` bit in the **EEPROM Done Status (EEDONE)** register until it is clear, indicating that the EEPROM has completed its power-on initialization. When `WORKING=0`, continue.
3. Read the `PRETRY` and `ERETRY` bits in the **EEPROM Support Control and Status (EESUPP)** register. If either of the bits are set, return an error, else continue.
4. Reset the EEPROM module using the **EEPROM Software Reset (SREEPROM)** register at offset 0x558 in the System Control register space.
5. Insert delay (6 cycles plus function call overhead).
6. Poll the `WORKING` bit in the **EEPROM Done Status (EEDONE)** register to determine when it is clear. When `WORKING=0`, continue.
7. Read the `PRETRY` and `ERETRY` bits in the **EESUPP** register. If either of the bits are set, return an error, else the EEPROM initialization is complete and software may use the peripheral as normal.

Important: Failure to perform these initialization steps after a reset may lead to incorrect operation or permanent data loss if the EEPROM is later written.

If the `PRETRY` or `ERETRY` bits are set in the **EESUPP** register, the EEPROM was unable to recover its state. If power is stable when this occurs, this indicates a fatal error and is likely an indication that the EEPROM memory has exceeded its specified lifetime write/erase specification. If the supply voltage is unstable when this return code is observed, retrying the operation once the voltage is stabilized may clear the error.

The EEPROM initialization function code is named EEPROMinit( ) in TivaWare, which can be downloaded from http://www.ti.com/tivaware.

7.3 Register Map

Table 7-3 on page 473 lists the ROM Controller register and the Flash memory and control registers. The offset listed is a hexadecimal increment to the particular memory controller's base address. The Flash memory register offsets are relative to the Flash memory control base address of 0x400F.D000. The EEPROM registers are relative to the EEPROM base address of 0x400A.F000. The ROM and Flash memory protection register offsets are relative to the System Control base address of 0x400F.E000.

Table 7-3. Flash Register Map

Offset	Name	Type	Reset	Description	See page
Flash Memory Registers (Flash Control Offset)					
0x000	FMA	RW	0x0000.0000	Flash Memory Address	475
0x004	FMD	RW	0x0000.0000	Flash Memory Data	476
0x008	FMC	RW	0x0000.0000	Flash Memory Control	477
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	479
0x010	FCIM	RW	0x0000.0000	Flash Controller Interrupt Mask	482
0x014	FCMISC	RW1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	484
0x020	FMC2	RW	0x0000.0000	Flash Memory Control 2	487
0x030	FWBVAL	RW	0x0000.0000	Flash Write Buffer Valid	488
0x100 - 0x17C	FWBn	RW	0x0000.0000	Flash Write Buffer n	489
0xFC0	FSIZE	RO	0x0000.000F	Flash Size	490
0xFC4	SSIZE	RO	0x0000.002F	SRAM Size	491
0xFCC	ROMSWMAP	RO	0x0000.0000	ROM Software Map	492
EEPROM Registers (EEPROM Control Offset)					
0x000	EESIZE	RO	0x0020.0200	EEPROM Size Information	493
0x004	EEBLOCK	RW	0x0000.0000	EEPROM Current Block	494
0x008	EEOFFSET	RW	0x0000.0000	EEPROM Current Offset	495
0x010	EERDWR	RW	-	EEPROM Read-Write	496
0x014	EERDWRINC	RW	-	EEPROM Read-Write with Increment	497
0x018	EEDONE	RO	0x0000.0000	EEPROM Done Status	498
0x01C	EESUPP	RW	-	EEPROM Support Control and Status	500
0x020	EEUNLOCK	RW	-	EEPROM Unlock	502
0x030	EEPROT	RW	0x0000.0000	EEPROM Protection	503
0x034	EEPASS0	RW	-	EEPROM Password	505
0x038	EEPASS1	RW	-	EEPROM Password	505
0x03C	EEPASS2	RW	-	EEPROM Password	505
0x040	EEINT	RW	0x0000.0000	EEPROM Interrupt	506
0x050	EEHIDE	RW	0x0000.0000	EEPROM Block Hide	507
0x080	EEDBGME	RW	0x0000.0000	EEPROM Debug Mass Erase	508
0xFC0	EEPROMPP	RO	0x0000.001F	EEPROM Peripheral Properties	509

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

Offset	Name	Type	Reset	Description	See page
Memory Registers (System Control Offset)					
0x0F0	RMCTL	RW1C	-	ROM Control	510
0x130	FMPRE0	RW	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	511
0x200	FMPRE0	RW	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	511
0x134	FMPPE0	RW	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	512
0x400	FMPPE0	RW	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	512
0x1D0	BOOTCFG	RO	0xFFFF.FFFE	Boot Configuration	513
0x1E0	USER_REG0	RW	0xFFFF.FFFF	User Register 0	516
0x1E4	USER_REG1	RW	0xFFFF.FFFF	User Register 1	516
0x1E8	USER_REG2	RW	0xFFFF.FFFF	User Register 2	516
0x1EC	USER_REG3	RW	0xFFFF.FFFF	User Register 3	516

7.4 Flash Memory Register Descriptions (Flash Control Offset)

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

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

During a single word write operation, this register contains a 4-byte-aligned address and specifies where the data is written. During a write operation that uses the write buffer, this register contains a 128-byte (32-word) aligned address that specifies the start of the 32-word block to be written. During erase operations, this register contains a 1 KB-aligned CPU byte address and specifies which block is erased. Note that the alignment requirements must be met by software or the results of the operation are unpredictable.

Flash Memory Address (FMA)

Base 0x400F.D000
Offset 0x000
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	OFFSET														
Type	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:15	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.
14:0	OFFSET	RW	0x0	Address Offset Address offset in Flash memory where operation is performed, except for non-volatile registers (see “Non-Volatile Register Programming” on page 465 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. This register is not used during erase cycles.

Flash Memory Data (FMD)

Base 0x400F.D000
Offset 0x004
Type RW, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:0	DATA	RW	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 475). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 476) 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 RW, 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	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	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. Depending on the value of the KEY bit in the **BOOTCFG** register, the value 0xA442 or 0x71D5 must be written into this field for a Flash memory write to occur. Writes to the **FMC** register without this WRKEY value are ignored. A read of this field returns the value 0.
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	COMT	RW	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 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. See “Non-Volatile Register Programming” on page 465 for more information on programming Flash-memory-resident registers.

Bit/Field	Name	Type	Reset	Description
2	MERASE	RW	0	Mass Erase Flash Memory This bit is used to mass erase the Flash main memory and to monitor the progress of that process. Value Description 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous mass erase operation is complete. 1 Set this bit to erase the Flash main memory. When read, a 1 indicates that the previous mass erase operation is not complete. For information on erase time, see “Flash Memory and EEPROM” on page 1146.
1	ERASE	RW	0	Erase a Page of Flash Memory This bit is used to erase a page of Flash memory and to monitor the progress of that process. Value Description 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous page erase operation is complete. 1 Set this bit to erase the Flash memory page specified by the contents of the **FMA** register. When read, a 1 indicates that the previous page erase operation is not complete. For information on erase time, see “Flash Memory and EEPROM” on page 1146.
0	WRITE	RW	0	Write a Word into Flash Memory This bit is used to write a word into Flash memory and to monitor the progress of that process. Value Description 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous write update operation is complete. 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. When read, a 1 indicates that the write update operation is not complete. For information on programming time, see “Flash Memory and EEPROM” on page 1146.

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		PROGRIS	reserved	ERRIS	INVDRIS	VOLTRIS	reserved						ERIS	PRIS	ARIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:14	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	PROGRIS	RO	0	Program Verify Error Raw Interrupt Status Value Description 0 An interrupt has not occurred. 1 An interrupt is pending because the verify of a PROGRAM operation failed. If this error occurs when using the Flash write buffer, software must inspect the affected words to determine where the error occurred. This bit is cleared by writing a 1 to the `PROGMISC` bit in the **FCMISC** register.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	ERRIS	RO	0	Erase Verify Error Raw Interrupt Status Value Description 0 An interrupt has not occurred. 1 An interrupt is pending because the verify of an ERASE operation failed. If this error occurs when using the Flash write buffer, software must inspect the affected words to determine where the error occurred. This bit is cleared by writing a 1 to the `ERMISC` bit in the **FCMISC** register.

Bit/Field	Name	Type	Reset	Description
10	INVDRIS	RO	0	Invalid Data Raw Interrupt Status Value Description 0 An interrupt has not occurred. 1 An interrupt is pending because a bit that was previously programmed as a 0 is now being requested to be programmed as a 1. This bit is cleared by writing a 1 to the `INVMISC` bit in the **FCMISC** register.
9	VOLTRIS	RO	0	Pump Voltage Raw Interrupt Status Value Description 0 An interrupt has not occurred. 1 An interrupt is pending because the regulated voltage of the pump went out of spec during the Flash operation and the operation was terminated. This bit is cleared by writing a 1 to the `VOLTMISC` bit in the **FCMISC** register.
8:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	ERIS	RO	0	EEPROM Raw Interrupt Status This bit provides status EEPROM operation. Value Description 0 An EEPROM interrupt has not occurred. 1 An EEPROM interrupt has occurred. This bit is cleared by writing a 1 to the `EMISC` bit in the **FCMISC** register.
1	PRIS	RO	0	Programming Raw Interrupt Status This bit provides status on programming cycles which are write or erase actions generated through the **FMC** or **FMC2** register bits (see page 477 and page 487). Value Description 0 The programming or erase cycle has not completed. 1 The programming or erase cycle has completed. This status is sent to the interrupt controller when the `PMASK` bit in the **FCIM** register is set. This bit is cleared by writing a 1 to the `PMISC` bit in the **FCMISC** register.

Bit/Field	Name	Type	Reset	Description
0	ARIS	RO	0	Access Raw Interrupt Status Value Description 0 No access has tried to improperly program or erase the Flash memory. 1 A program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the **FMPPEn** registers. This status is sent to the interrupt controller when the `AMASK` bit in the **FCIM** register is set. This bit is cleared by writing a 1 to the `AMISC` bit in the **FCMISC** register.

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

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

Flash Controller Interrupt Mask (FCIM)

Base 0x400F.D000
Offset 0x010
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PROGMASK	reserved	ERMASK	INVDMASK	VOLTMASK	reserved						EMASK	PMASK	AMASK
Type	RO	RO	RW	RO	RW	RW	RW	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:14	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	PROGMASK	RW	0	PROGVER Interrupt Mask Value Description 0 The `PROGRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `PROGRIS` bit is set.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	ERMASK	RW	0	ERVER Interrupt Mask Value Description 0 The `ERRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `ERRIS` bit is set.
10	INVDMASK	RW	0	Invalid Data Interrupt Mask Value Description 0 The `INVDRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `INVDRIS` bit is set.

Bit/Field	Name	Type	Reset	Description
9	VOLTMASK	RW	0	VOLT Interrupt Mask Value Description 0 The VOLTRIS interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the VOLTRIS bit is set.
8:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	EMASK	RW	0	EEPROM Interrupt Mask Value Description 0 The ERIS interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the ERIS bit is set.
1	PMASK	RW	0	Programming Interrupt Mask This bit controls the reporting of the programming raw interrupt status to the interrupt controller. Value Description 0 The PRIS interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the PRIS bit is set.
0	AMASK	RW	0	Access Interrupt Mask This bit controls the reporting of the access raw interrupt status to the interrupt controller. Value Description 0 The ARIS interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the ARIS bit is set.

Register 6: Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014

This register provides two functions. First, it reports the cause of an interrupt by indicating which interrupt source or sources are signalling the interrupt. Second, it serves as the method to clear the interrupt reporting.

Flash Controller Masked Interrupt Status and Clear (FCMISC)

Base 0x400F.D000
Offset 0x014
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PROGMISC	reserved	ERMISC	INVDMISC	VOLTMISC	reserved						EMISC	PMISC	AMISC
Type	RO	RO	RW1C	RO	RW1C	RW1C	RW1C	RO	RO	RO	RO	RO	RO	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
31:14	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	PROGMISC	RW1C	0	PROGVER Masked Interrupt Status and Clear Value Description 0 When read, a 0 indicates that an interrupt has not occurred. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled. Writing a 1 to this bit clears `PROGMISC` and also the `PROGRIS` bit in the **FCRIS** register (see page 479).
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	ERMISC	RW1C	0	ERVER Masked Interrupt Status and Clear Value Description 0 When read, a 0 indicates that an interrupt has not occurred. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled. Writing a 1 to this bit clears `ERMISC` and also the `ERRIS` bit in the **FCRIS** register (see page 479).

Bit/Field	Name	Type	Reset	Description
10	INVDMISC	RW1C	0	Invalid Data Masked Interrupt Status and Clear Value Description 0 When read, a 0 indicates that an interrupt has not occurred. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled. Writing a 1 to this bit clears `INVDMISC` and also the `INVDRIS` bit in the **FCRIS** register (see page 479).
9	VOLTMISC	RW1C	0	VOLT Masked Interrupt Status and Clear Value Description 0 When read, a 0 indicates that an interrupt has not occurred. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled. Writing a 1 to this bit clears `VOLTMISC` and also the `VOLTRIS` bit in the **FCRIS** register (see page 479).
8:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	EMISC	RW1C	0	EEPROM Masked Interrupt Status and Clear Value Description 0 When read, a 0 indicates that an interrupt has not occurred. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled. Writing a 1 to this bit clears `EMISC` and also the `ERIS` bit in the **FCRIS** register (see page 479).
1	PMISC	RW1C	0	Programming Masked Interrupt Status and Clear Value Description 0 When read, a 0 indicates that a programming cycle complete interrupt has not occurred. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled because a programming cycle completed. Writing a 1 to this bit clears `PMISC` and also the `PRIS` bit in the **FCRIS** register (see page 479).

Bit/Field	Name	Type	Reset	Description
0	AMISC	RW1C	0	Access Masked Interrupt Status and Clear Value Description 0 When read, a 0 indicates that no improper accesses have occurred. A write of 0 has no effect on the state of this bit. 1 When read, a 1 indicates that an unmasked interrupt was signaled because a program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the **FMPPEn** registers. Writing a 1 to this bit clears `AMISC` and also the `ARIS` bit in the **FCRIS** register (see page 479).

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 475). If the access is a write access, the data contained in the **Flash Write Buffer (FWB)** registers is written.

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

Flash Memory Control 2 (FMC2)

Base 0x400F.D000
Offset 0x020
Type RW, reset 0x0000.0000

	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	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	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. Depending on the value of the KEY bit in the **BOOTCFG** register, the value 0xA442 or 0x71D5 must be written into this field for a Flash memory 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	RW	0	Buffered Flash Memory Write This bit is used to start a buffered write to Flash memory. Value Description 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. 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. For information on programming time, see “Flash Memory and EEPROM” on page 1146.

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

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

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

Flash Write Buffer Valid (FWBVAL)

Base 0x400F.D000
Offset 0x030
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FWB[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FWB[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	FWB[n]	RW	0x0	Flash Memory Write Buffer Value Description 0 The corresponding **FWBn** register has no new data to be written. 1 The corresponding **FWBn** register has been updated since the last buffer write operation and is ready to be written to Flash memory. Bit 0 corresponds to **FWB0**, offset 0x100, and bit 31 corresponds to **FWB31**, offset 0x13C.

Register 9: Flash Write Buffer n (FWBn), offset 0x100 - 0x17C

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

Flash Write Buffer n (FWBn)

Base 0x400F.D000
Offset 0x100 - 0x17C
Type RW, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:0	DATA	RW	0x0000.0000	Data Data to be written into the Flash memory.

Register 10: Flash Size (FSIZE), offset 0xFC0

This register indicates the size of the on-chip Flash memory.

Important: This register should be used to determine the size of the Flash memory that is implemented on this microcontroller. However, to support legacy software, the **DC0** register is available. A read of the **DC0** register correctly identifies legacy memory sizes. Software must use the **FSIZE** register for memory sizes that are not listed in the **DC0** register description.

Flash Size (FSIZE)

Base 0x400F.D000
Offset 0xFC0
Type RO, reset 0x0000.000F

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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	0	0	0	0	0	0	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	SIZE	RO	0xF	Flash Size Indicates the size of the on-chip Flash memory.

Value	Description
0x000F	32 KB of Flash

Register 11: SRAM Size (SSIZE), offset 0xFC4

This register indicates the size of the on-chip SRAM.

Important: This register should be used to determine the size of the SRAM that is implemented on this microcontroller. However, to support legacy software, the **DC0** register is available. A read of the **DC0** register correctly identifies legacy memory sizes. Software must use the **SSIZE** register for memory sizes that are not listed in the **DC0** register description.

SRAM Size (SSIZE)

Base 0x400F.D000
Offset 0xFC4
Type RO, reset 0x0000.002F

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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	0	0	0	0	1	0	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	SIZE	RO	0x2F	SRAM Size Indicates the size of the on-chip SRAM.

Value	Description
0x002F	12 KB of SRAM

Register 12: ROM Software Map (ROMSWMAP), offset 0xFCC

This register indicates the presence of third-party software in the on-chip ROM.

Important: This register should be used to determine the presence of third-party software in the on-chip ROM on this microcontroller. However, to support legacy software, the **NVMSTAT** register is available. A read of the TPSW bit in the **NVMSTAT** register correctly identifies the presence of legacy third-party software. Software should use the **ROMSWMAP** register for software that is not on legacy devices.

ROM Software Map (ROMSWMAP)

Base 0x400F.D000
Offset 0xFCC
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															SAFERTOS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	Software should not rely on the value of 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	SAFERTOS	RO	0x0	SafeRTOS Present

Value	Description
0	SafeRTOS is not in the on-chip ROM.
1	SafeRTOS is in the on-chip ROM.

7.5 EEPROM Register Descriptions (EEPROM Offset)

This section lists and describes the EEPROM registers, in numerical order by address offset. Registers in this section are relative to the EEPROM base address of 0x400A.F000.

Note that the EEPROM module clock must be enabled before the registers can be programmed (see page 332). There must be a delay of 3 system clocks after the EEPROM module clock is enabled before any EEPROM module registers are accessed. In addition, after enabling or resetting the EEPROM module, software must wait until the WORKING bit in the **EEDONE** register is clear before accessing any EEPROM registers.

Register 13: EEPROM Size Information (EESIZE), offset 0x000

The **EESIZE** register indicates the number of 16-word blocks and 32-bit words in the EEPROM.

EEPROM Size Information (EESIZE)

Base 0x400A.F000
Offset 0x000
Type RO, reset 0x0020.0200

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved					BLKCNT										
Type	RO	RO	RO	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	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WORDCNT															
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
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26:16	BLKCNT	RO	0x20	Number of 16-Word Blocks This value encoded in this field describes the number of 16-word blocks in the EEPROM.
15:0	WORDCNT	RO	0x200	Number of 32-Bit Words This value encoded in this field describes the number of 32-bit words in the EEPROM.

Register 14: EEPROM Current Block (EEBLOCK), offset 0x004

The **EEBLOCK** register is used to select the EEPROM block for subsequent reads, writes, and protection control. The value is a block offset into the EEPROM, such that the first block is 0, then second block is 1, etc. Each block contains 16 words. Attempts to set an invalid block causes the BLOCK field to be configured to 0. To verify that the intended block is being accessed, software can read the BLOCK field after it has been written. An invalid block can be either a non-existent block or a block that has been hidden using the **EEHIDE** register. Note that block 0 cannot be hidden.

EEPROM Current Block (EEBLOCK)

Base 0x400A.F000
Offset 0x004
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	BLOCK	RW	0x0000	Current Block This field specifies the block in the EEPROM that is selected for subsequent accesses. Once this field is configured, the read-write registers operate against the specified block, using the **EEOFFSET** register to select the word within the block. Additionally, the protection and unlock registers are used for the selected block. The maximum value that can be written into this register is determined by the block count, as indicated by the **EESIZE** register. Attempts to write this field larger than the maximum number of blocks or to a locked block causes this field to be configured to 0.

Register 15: EEPROM Current Offset (EEOFFSET), offset 0x008

The **EEOFFSET** register is used to select the EEPROM word to read or write within the block selected by the **EEBLOCK** register. The value is a word offset into the block. Because accesses to the **EERDWRINC** register change the offset, software can read the contents of this register to determine the current offset.

EEPROM Current Offset (EEOFFSET)

Base 0x400A.F000
Offset 0x008
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												OFFSET			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	OFFSET	RW	0x0	Current Address Offset This value is the current address specified as an offset into the block selected by the **EEBLOCK** register. Once configured, the read-write registers, **EERDWR** and **EERDWRINC**, operate against that address. The offset is automatically incremented by the **EERDWRINC** register, with wrap around within the block, which means the offset is incremented from 15 back to 0.

Register 16: EEPROM Read-Write (EERDWR), offset 0x010

The **EERDWR** register is used to read or write the EEPROM word at the address pointed to by the **EEBLOCK** and **EEOFFSET** registers. If the protection or access rules do not permit access, the operation is handled as follows: if reading is not allowed, the value 0xFFFF.FFFF is returned in all cases; if writing is not allowed, the **EEDONE** register is configured to indicate an error.

EEPROM Read-Write (EERDWR)

Base 0x400A.F000
Offset 0x010
Type RW, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:0	VALUE	RW	-	EEPROM Read or Write Data On a read, this field contains the value at the word pointed to by **EEOFFSET**. On a write, this field contains the data to be stored at the word pointed to by **EEOFFSET**. For writes, configuring this field starts the write process. If protection and access rules do not permit reads, all 1s are returned. If protection and access rules do not permit writes, the write fails and the **EEDONE** register indicates failure.

Register 17: EEPROM Read-Write with Increment (EERDWRINC), offset 0x014

The **EERDWRINC** register is used to read or write the EEPROM word at the address pointed to by the **EEBLOCK** and **EEOFFSET** registers, and then increment the OFFSET field in the **EEOFFSET** register. If the protection or access rules do not permit access, the operation is handled as follows: if reading is not allowed, the value 0xFFFF.FFFF is returned in all cases; if writing is not allowed, the **EEDONE** register is configured to indicate an error. In all cases, the OFFSET field is incremented. If the last value is reached, OFFSET wraps around to 0 and points to the first word.

EEPROM Read-Write with Increment (EERDWRINC)

Base 0x400A.F000
Offset 0x014
Type RW, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:0	VALUE	RW	-	EEPROM Read or Write Data with Increment On a read, this field contains the value at the word pointed to by **EEOFFSET**. On a write, this field contains the data to be stored at the word pointed to by **EEOFFSET**. For writes, configuring this field starts the write process. If protection and access rules do not permit reads, all 1s are returned. If protection and access rules do not permit writes, the write fails and the **EEDONE** register indicates failure. Regardless of error, the OFFSET field in the **EEOFFSET** register is incremented by 1, and the value wraps around if the last word is reached.

Register 18: EEPROM Done Status (EEDONE), offset 0x018

The **EEDONE** register indicates the successful or failed completion of a write using the **EERDWR** or **EERDWRINC** register, protection set using the **EEPROT** register, password registered using the **EEPASS** register, copy buffer erase or program retry using the **EESUPP** register, or a debug mass erase using the **EEDBGME** register. The **EEDONE** register can be used with the **EEINT** register to generate an interrupt to report the status. The normal usage is to poll the **EEDONE** register or read the register after an interrupt is triggered. When the **EEDONE** bit 0 is set, then the operation is still in progress. When the **EEDONE** bit 0 is clear, then the value of **EEDONE** indicates the completion status. If **EEDONE**==0, then the write completed successfully. If **EEDONE**!=0, then an error occurred and the source of the error is given by the set bit(s). If an error occurs, corrective action may be taken as explained on page 500.

EEPROM Done Status (EEDONE)

Base 0x400A.F000
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										WRBUSY	NOPERM	WKCOPY	WKERASE	reserved	WORKING
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	WRBUSY	RO	0	Write Busy Value Description 0 No error 1 An attempt to access the EEPROM was made while a write was in progress.
4	NOPERM	RO	0	Write Without Permission Value Description 0 No error 1 An attempt was made to write without permission. This error can result because the block is locked, the write violates the programmed access protection, or when an attempt is made to write a password when the password has already been written.

Bit/Field	Name	Type	Reset	Description
3	WKCOPY	RO	0	Working on a Copy Value Description 0 The EEPROM is not copying. 1 A write is in progress and is waiting for the EEPROM to copy to or from the copy buffer.
2	WKERASE	RO	0	Working on an Erase Value Description 0 The EEPROM is not erasing. 1 A write is in progress and the original block is being erased after being copied.
1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WORKING	RO	0	EEPROM Working Value Description 0 The EEPROM is not working. 1 The EEPROM is performing the requested operation.

Register 19: EEPROM Support Control and Status (EESUPP), offset 0x01C

The **EESUPP** register indicates if internal operations are required because an internal copy buffer must be erased or a programming failure has occurred and the operation must be completed. These conditions are explained below as well as in more detail in the section called “Manual Copy Buffer Erase” on page 470 and the section called “Error During Programming” on page 470.

- The EREQ bit is set if the internal copy buffer must be erased the next time it is used because it is full. To avoid the delay of waiting for the copy buffer to be erased on the next write, it can be erased manually using this register by setting the START bit.
- If either PRETRY or ERETRY is set indicating that an operation must be completed, setting the START bit causes the operation to be performed again.
- The PRETRY and ERETRY bits are cleared automatically after the failed operation has been successfully completed.

These bits are not changed by reset, so any condition that occurred before a reset is still indicated after a reset.

EEPROM Support Control and Status (EESUPP)

Base 0x400A.F000
Offset 0x01C
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												PRETRY	ERETRY	EREQ	START
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	PRETRY	RO	-	Programming Must Be Retried
2	ERETRY	RO	-	Erase Must Be Retried

PRETRY:

Value	Description
0	Programming has not failed.
1	Programming from a copy in either direction failed to complete and must be restarted by setting the START bit.

ERETRY:

Value	Description
0	Erasing has not failed.
1	Erasing failed to complete and must be restarted by setting the START bit. If the failed erase is due to the erase of a main buffer, the copy will be performed after the erase completes successfully.

Bit/Field	Name	Type	Reset	Description
1	EREQ	RO	-	Erase Required Value Description 0 The copy buffer has available space. 1 An erase of the copy buffer is required.
0	START	RW	0	Start Erase Setting this bit starts error recovery if the PRETRY or ERETRY bit is set. If both the PRETRY and the ERETRY bits are clear, setting this bit starts erasing the copy buffer if EREQ is set. If none of the other bits in this register are set, setting this bit is ignored. After this bit is set, the WORKING bit in the **EEDONE** register is set and is cleared when the operation is complete. In addition, the **EEINT** register can be used to generate an interrupt on completion. If this bit is set while an operation is in progress, the write is ignored. The START bit is automatically cleared when the operation completes.

Register 20: EEPROM Unlock (EEUNLOCK), offset 0x020

The **EEUNLOCK** register can be used to unlock the whole EEPROM or a single block using a password. Unlocking is only required if a password is registered using the **EEPASSn** registers for the block that is selected by the **EEBLOCK** register. If block 0 has a password, it locks the remaining blocks from any type of access, but uses its own protection mechanism, for example readable, but not writable when locked. In addition, if block 0 has a password, it must be unlocked before unlocking any other block.

The **EEUNLOCK** register is written between 1 and 3 times to form the 32-bit, 64-bit, or 96-bit password registered using the **EEPASSn** registers. The value used to configure the **EEPASS0** register must always be written last. For example, for a 96-bit password, the value used to configure the **EEPASS2** register must be written first followed by the **EEPASS1** and **EEPASS0** register values. The block or the whole EEPROM can be re-locked by writing 0xFFFF.FFFF to this register.

In the event that an invalid value is written to this register, the block remains locked. The state of the EEPROM lock can be determined by reading back the **EEUNLOCK** register. If a multi-word password is set and the number of words written is incorrect, writing 0xFFFF.FFFF to this register reverts the EEPROM lock to the locked state, and the proper unlock sequence can be retried.

Note that the internal logic is balanced to prevent any electrical or time-based attack being used to find the correct password or its length.

EEPROM Unlock (EEUNLOCK)

Base 0x400A.F000
Offset 0x020
Type RW, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:0	UNLOCK	RW	-	EEPROM Unlock Value Description 0 The EEPROM is locked. 1 The EEPROM is unlocked. The EEPROM is locked if the block referenced by the **EEBLOCK** register has a password registered, or if the master block (block 0) has a password. Unlocking is performed by writing the password to this register. The block or the EEPROM stays unlocked until it is locked again or until the next reset. It can be locked again by writing 0xFFFF.FFFF to this register.

Register 21: EEPROM Protection (EEPROT), offset 0x030

The **EEPROT** register is used to set or read the protection for the current block, as selected by the **EEBLOCK** register. Protection and access control is used to determine when a block's contents can be read or written. The protection level for block 0 sets the minimum protection level for the entire EEPROM. For example, if the PROT field is configured to 0x1 for block 0, then block 1 could be configured with the PROT field to be 0x1, 0x2, or 0x3, but not 0x0.

EEPROM Protection (EEPROT)

Base 0x400A.F000
Offset 0x030
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												ACC	PROT		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ACC	RW	0	Access Control

Value	Description
0	Both user and supervisor code may access this block of the EEPROM.
1	Only supervisor code may access this block of the EEPROM. µDMA and Debug are also prevented from accessing the EEPROM.

If this bit is set for block 0, then the whole EEPROM may only be accessed by supervisor code.

Bit/Field	Name	Type	Reset	Description
2:0	PROT	RW	0x0	Protection Control The Protection bits control what context is needed for reading and writing the block selected by the **EEBLOCK** register, or if block 0 is selected, all blocks. The following values are allowed: Value Description 0x0 This setting is the default. If there is no password, the block is not protected and is readable and writable. If there is a password, the block is readable, but only writable when unlocked. 0x1 If there is a password, the block is readable or writable only when unlocked. This value has no meaning when there is no password. 0x2 If there is no password, the block is readable, not writable. If there is a password, the block is readable only when unlocked, but is not writable under any conditions. 0x3 Reserved

Register 22: EEPROM Password (EEPASS0), offset 0x034

Register 23: EEPROM Password (EEPASS1), offset 0x038

Register 24: EEPROM Password (EEPASS2), offset 0x03C

The **EEPASSn** registers are used to configure a password for a block. A password may only be set once and cannot be changed. The password may be 32-bits, 64-bits, or 96-bits. Each word of the password can be any 32-bit value other than 0xFFFF.FFFF (all 1s). To set a password, the **EEPASS0** register is written to with a value other than 0xFFFF.FFFF. When the write completes, as indicated in the **EEDONE** register, the application may choose to write to the **EEPASS1** register with a value other than 0xFFFF.FFFF. When that write completes, the application may choose to write to the **EEPASS2** register with a value other than 0xFFFF.FFFF to create a 96-bit password. The registers do not have to be written consecutively, and the **EEPASS1** and **EEPASS2** registers may be written at a later date. Based on whether 1, 2, or all 3 registers have been written, the unlock code also requires the same number of words to unlock.

Note: Once the password is written, the block is not actually locked until either a reset occurs or 0xFFFF.FFFF is written to **EEUNLOCK**.

EEPROM Password (EEPASSn)

Base 0x400A.F000
Offset 0x034
Type RW, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:0	PASS	RW	-	Password This register reads as 0x1 if a password is registered for this block and 0x0 if no password is registered. A write to this register if it reads as 0x0 sets the password. If an attempt is made to write to this register when it reads as 0x1, the write is ignored and the `NOPERM` bit in the **EEDONE** register is set.

Register 25: EEPROM Interrupt (EEINT), offset 0x040

The **EEINT** register is used to control whether an interrupt should be generated when a write to EEPROM completes as indicated by the **EEDONE** register value changing from 0x1 to any other value. If the INT bit in this register is set, the ERIS bit in the **Flash Controller Raw Interrupt Status (FCRIS)** register is set whenever the **EEDONE** register value changes from 0x1 as the Flash memory and the EEPROM share an interrupt vector.

EEPROM Interrupt (EEINT)

Base 0x400A.F000
Offset 0x040
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

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

Value	Description
0	No interrupt is generated.
1	An interrupt is generated when the **EEDONE** register transitions from 1 to 0 or an error occurs. The **EEDONE** register provides status after a write to an offset location as well as a write to the password and protection bits.

Register 26: EEPROM Block Hide (EEHIDE), offset 0x050

The **EEHIDE** register is used to hide one or more blocks other than block 0. Once hidden, the block is not accessible until the next reset. This model allows initialization code to have access to data which is not visible to the rest of the application. This register also provides for additional security in that there is no password to search for in the code or data.

EEPROM Block Hide (EEHIDE)

Base 0x400A.F000
Offset 0x050
Type RW, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Hn															reserved
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	Hn	RW	0x0000.000	Hide Block Value Description 0 The corresponding block is not hidden. 1 The block number that corresponds to the bit number is hidden. A hidden block cannot be accessed, and the OFFSET value in the **EEBLOCK** register cannot be set to that block number. If an attempt is made to configure the OFFSET field to a hidden block, the **EEBLOCK** register is cleared. Any attempt to clear a bit in this register that is set is ignored.
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 27: EEPROM Debug Mass Erase (EEDBGME), offset 0x080

The **EEDBGME** register is used to mass erase the EEPROM block back to its default state from the factory. This register is intended to be used only for debug and test purposes, not in production environments. The erase takes place in such a way as to be secure. It first erases all data and then erases the protection mechanism. This register can only be written from supervisor mode by the core, and can also be written by the TM4C1232C3PM debug controller when enabled. A key is used to avoid accidental use of this mechanism. Note that if a power down takes place while erasing, the mechanism should be used again to complete the operation. Powering off prematurely does not expose secured data.

To start a mass erase, the whole register must be written as 0xE37B.0001. The register reads back as 0x1 until the erase is fully completed at which time it reads as 0x0. The **EEDONE** register is set to 0x1 when the erase is started and changes to 0x0 or an error when the mass erase is complete.

Note that mass erasing the EEPROM block means that the wear-leveling counters are also reset to the factory default.

EEPROM Debug Mass Erase (EEDBGME)

Base 0x400A.F000
Offset 0x080
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	KEY															
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															ME
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	KEY	WO	0x0000	Erase Key This field must be written with 0xE37B for the `ME` field to be effective.
15:1	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ME	RW	0	Mass Erase

Value	Description
0	No action.
1	When written as a 1, the EEPROM is mass erased. This bit continues to read as 1 until the EEPROM is fully erased.

Register 28: EEPROM Peripheral Properties (EEPROMPP), offset 0xFC0

The **EEPROMPP** register indicates the size of the EEPROM for this part.

EEPROM Peripheral Properties (EEPROMPP)

Base 0x400A.F000
Offset 0xFC0
Type RO, reset 0x0000.001F

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											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	0	0	0	0	0	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	SIZE	RO	0x1F	2-KB EEPROM Size

7.6 Memory Register Descriptions (System Control Offset)

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

Register 29: ROM Control (RMCTL), offset 0x0F0

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

At reset, the following sequence is performed:

1. The **BOOTCFG** register is read. If the EN bit is clear, the ROM Boot Loader is executed.
2. In the ROM Boot Loader, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
3. If the EN bit is set or the status doesn't match the specified polarity, the data at address 0x0000.0004 is read, and if the data at this address is 0xFFFF.FFFF, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
4. If there is data at address 0x0000.0004 that is not 0xFFFF.FFFF, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

ROM Control (RMCTL)

Base 0x400F.E000
Offset 0x0F0
Type RW1C, 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	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	BA	RW1C	1	Boot Alias Value Description 0 The Flash memory is at address 0x0. 1 The microcontroller's ROM appears at address 0x0. This bit is cleared by writing a 1 to this bit position.

Register 30: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200

Note: The **FMPRE0** register is aliased for backwards compatibility.

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits).

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 2-KB blocks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is RW0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 194.

Each **FMPREn** register controls a 64-k block of Flash. For additional information, see “Flash Memory Protection” on page 461.

- **FMPRE0**: 0 to 64 KB

Flash Memory Protection Read Enable n (FMPREn)

Base 0x400F.E000
Offset 0x130 and 0x200
Type RW, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	READ_ENABLE															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	RW	0xFFFF.FFFF	Flash Read Enable Each bit configures a 2-KB flash block to be read only. The policies may be combined as shown in Table 7-1 on page 462.

Register 31: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400

Note: The **FMPPE0** register is aliased for backwards compatibility.

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the read-only protection bits).

This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is RW0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 194. For additional information, see “Flash Memory Protection” on page 461.

Each **FMPPEn** register controls a 64-k block of Flash. For additional information, see “Flash Memory Protection” on page 461.

- **FMPPE0**: 0 to 64 KB

Flash Memory Protection Program Enable n (FMPPEn)

Base 0x400F.E000
Offset 0x134 and 0x400
Type RW, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PROG_ENABLE															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	RW	0xFFFF.FFFF	Flash Programming Enable Each bit configures a 2-KB flash block to be execute only. The policies may be combined as shown in Table 7-1 on page 462.

Register 32: Boot Configuration (BOOTCFG), offset 0x1D0

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

Note: The **Boot Configuration (BOOTCFG)** register requires a POR before the committed changes take effect.

This register is not written directly, but instead uses the **FMD** register as explained in "Non-Volatile Register Programming" on page 465. This register provides configuration of a GPIO pin to enable the ROM Boot Loader as well as a write-once mechanism to disable external debugger access to the device. At reset, the user has the opportunity to direct the core to execute the ROM Boot Loader or the application in Flash memory by using any GPIO signal from Ports A-Q as configured by the bits in this register. At reset, the following sequence is performed:

1. The **BOOTCFG** register is read. If the EN bit is clear, the ROM Boot Loader is executed.
2. In the ROM Boot Loader, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
3. If the EN bit is set or the status doesn't match the specified polarity, the data at address 0x0000.0004 is read, and if the data at this address is 0xFFFF.FFFF, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
4. If there is data at address 0x0000.0004 that is not 0xFFFF.FFFF, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

The DBG0 bit is cleared by the factory and the DBG1 bit is set, which enables external debuggers. Clearing the DBG1 bit disables any external debugger access to the device, starting with the next power-up cycle of the device. The NW bit indicates that bits in the register can be changed from 1 to 0.

By committing the register values using the COMT bit in the **FMC** register, the register contents become non-volatile and are therefore retained following power cycling. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset when the register is not yet committed; any other type of reset does not affect this register. Once committed, the register retains its value through power-on reset. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in "Recovering a "Locked" Microcontroller" on page 194.

Boot Configuration (BOOTCFG)

Base 0x400F.E000
Offset 0x1D0
Type RO, reset 0xFFFF.FFFE

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NW	reserved														
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
	PORT			PIN			POL	EN	reserved			KEY	reserved		DBG1	DBG0
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	0

Bit/Field	Name	Type	Reset	Description
31	NW	RO	1	Not Written When set, this bit indicates that the values in this register can be changed from 1 to 0. When clear, this bit specifies that the contents of this register cannot be changed.
30:16	reserved	RO	0xFFFF	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:13	PORT	RO	0x7	Boot GPIO Port This field selects the port of the GPIO port pin that enables the ROM boot loader at reset. Value Description 0x0 Port A 0x1 Port B 0x2 Port C 0x3 Port D 0x4 Port E 0x5 Port F 0x6 Port G 0x7 Port H
12:10	PIN	RO	0x7	Boot GPIO Pin This field selects the pin number of the GPIO port pin that enables the ROM boot loader at reset. Value Description 0x0 Pin 0 0x1 Pin 1 0x2 Pin 2 0x3 Pin 3 0x4 Pin 4 0x5 Pin 5 0x6 Pin 6 0x7 Pin 7
9	POL	RO	1	Boot GPIO Polarity When set, this bit selects a high level for the GPIO port pin to enable the ROM boot loader at reset. When clear, this bit selects a low level for the GPIO port pin.
8	EN	RO	1	Boot GPIO Enable Clearing this bit enables the use of a GPIO pin to enable the ROM Boot Loader at reset. When this bit is set, the contents of address 0x0000.0004 are checked to see if the Flash memory has been programmed. If the contents are not 0xFFFF.FFFF, the core executes out of Flash memory. If the Flash has not been programmed, the core executes out of ROM.

Bit/Field	Name	Type	Reset	Description
7:5	reserved	RO	0x7	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	KEY	RO	1	KEY Select This bit chooses between using the value 0xA442 or 0x71D5 as the WRKEY value in the **FMC/FMC2** register. Value Description 0 The value 0x71D5 is used as the WRKEY in the **FMC/FMC2** register. Writes to the **FMC/FMC2** register with a 0xA442 key are ignored. 1 0xA442 is used as the WRKEY in the **FMC/FMC2** register. Writes to the**FMC/FMC2** register with a 0x71D5 key are ignored.
3:2	reserved	RO	0x3	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	DBG1	RO	1	Debug Control 1 The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.
0	DBG0	RO	0	Debug Control 0 The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.

Register 33: User Register 0 (USER_REG0), offset 0x1E0

Register 34: User Register 1 (USER_REG1), offset 0x1E4

Register 35: User Register 2 (USER_REG2), offset 0x1E8

Register 36: User Register 3 (USER_REG3), offset 0x1EC

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

These registers each provide 32 bits of user-defined data that is non-volatile. Bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset when the register is not yet committed; any other type of reset does not affect this register. Once committed, the register retains its value through power-on reset. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 194.

User Register n (USER_REGn)

Base 0x400F.E000
Offset 0x1E0
Type RW, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DATA															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	DATA	RW	0xFFFF.FFFF	User Data Contains the user data value. This field is initialized to all 1s and once committed, retains its value through power-on reset.

8 Micro Direct Memory Access (μDMA)

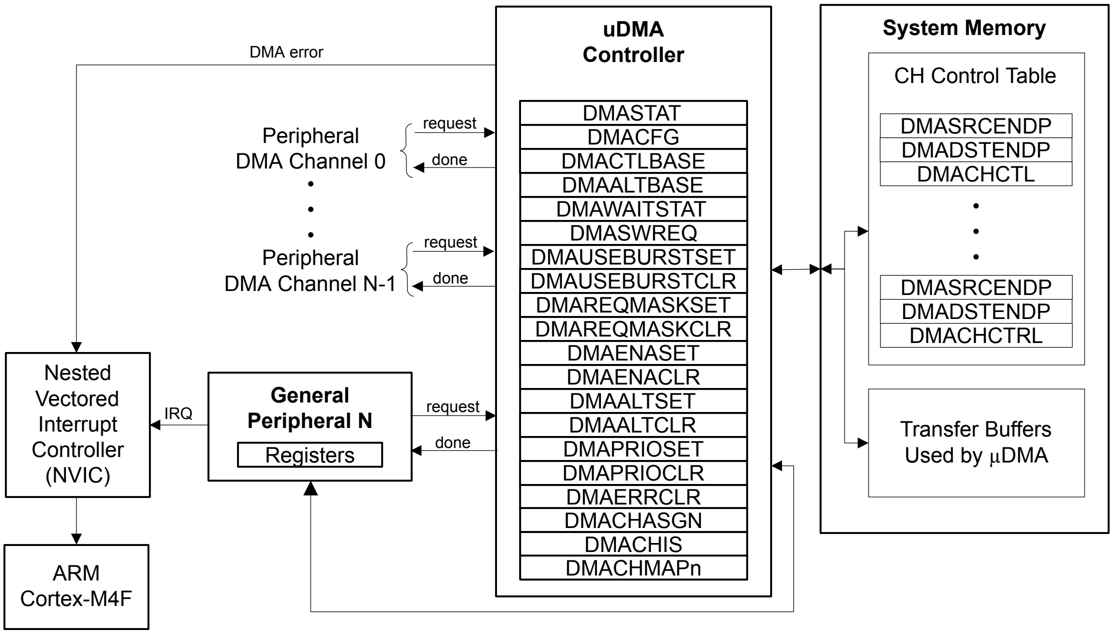
The TM4C1232C3PM microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (μDMA). The μDMA controller provides a way to offload data transfer tasks from the Cortex™-M4F processor, allowing for more efficient use of the processor and the available bus bandwidth. The μDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The μDMA controller provides the following features:

- ARM® PrimeCell® 32-channel configurable μDMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of up to 256 arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
  - Independently configured and operated channels
  - Dedicated channels for supported on-chip modules
  - Flexible channel assignments
  - One channel each for receive and transmit path for bidirectional modules
  - Dedicated channel for software-initiated transfers
  - Per-channel configurable priority scheme
  - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between μDMA controller and the processor core
  - μDMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Transfer size is programmable in binary steps from 1 to 1024
- Source and destination address increment size of byte, half-word, word, or no increment

- Maskable peripheral requests
- Interrupt on transfer completion, with a separate interrupt per channel

8.1 Block Diagram

Figure 8-1. μDMA Block Diagram



8.2 Functional Description

The μDMA controller is a flexible and highly configurable DMA controller designed to work efficiently with the microcontroller's Cortex-M4F processor core. It supports multiple data sizes and address increment schemes, multiple levels of priority among DMA channels, and several transfer modes to allow for sophisticated programmed data transfers. The μDMA controller's usage of the bus is always subordinate to the processor core, so it never holds up a bus transaction by the processor. Because the μDMA controller is only using otherwise-idle bus cycles, the data transfer bandwidth it provides is essentially free, with no impact on the rest of the system. The bus architecture has been optimized to greatly enhance the ability of the processor core and the μDMA controller to efficiently share the on-chip bus, thus improving performance. The optimizations include RAM striping and peripheral bus segmentation, which in many cases allow both the processor core and the μDMA controller to access the bus and perform simultaneous data transfers.

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

Each peripheral function that is supported has a dedicated channel on the μDMA controller that can be configured independently. The μDMA controller implements a unique configuration method using channel control structures that are maintained in system memory by the processor. While simple transfer modes are supported, it is also possible to build up sophisticated "task" lists in memory that allow the μDMA controller to perform arbitrary-sized transfers to and from arbitrary locations as part of a single transfer request. The μDMA controller also supports the use of ping-pong buffering to accommodate constant streaming of data to or from a peripheral.

Each channel also has a configurable arbitration size. The arbitration size is the number of items that are transferred in a burst before the μDMA controller re-arbitrates for channel priority. Using the arbitration size, it is possible to control exactly how many items are transferred to or from a peripheral each time it makes a μDMA service request.

8.2.1 Channel Assignments

Each DMA channel has up to five possible assignments which are selected using the **DMA Channel Map Select n (DMACHMAPn)** registers with 4-bit assignment fields for each μDMA channel.

Table 8-1 on page 519 shows the μDMA channel mapping. The Enc. column shows the encoding for the respective **DMACHMAPn** bit field. Encodings 0x5 - 0xF are all reserved. To support legacy software which uses the **DMA Channel Assignment (DMACHASGN)** register, Enc. 0 is equivalent to a **DMACHASGN** bit being clear, and Enc. 1 is equivalent to a **DMACHASGN** bit being set. If the **DMACHASGN** register is read, bit fields return 0 if the corresponding **DMACHMAPn** register field value are equal to 0, otherwise they return 1 if the corresponding **DMACHMAPn** register field values are not equal to 0. The Type indication in the table indicates if a particular peripheral uses a single request (S), burst request (B) or either (SB).

Note: Channels noted in the table as "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 μDMA channels 0-3 can be changed with the **USBDMASEL** register (see page 1080).

Table 8-1. μDMA Channel Assignments

Enc.	0		1		2		3		4	
Ch #	Peripheral	Type	Peripheral	Type	Peripheral	Type	Peripheral	Type	Peripheral	Type
0	USB0 EP1 RX	SB	UART2 RX	SB	Software	B	GPTimer 4A	B	Software	B
1	USB0 EP1 TX	B	UART2 TX	SB	Software	B	GPTimer 4B	B	Software	B
2	USB0 EP2 RX	B	GPTimer 3A	B	Software	B	Software	B	Software	B
3	USB0 EP2 TX	B	GPTimer 3B	B	Software	B	Software	B	Software	B
4	USB0 EP3 RX	B	GPTimer 2A	B	Software	B	GPIO A	B	Software	B
5	USB0 EP3 TX	B	GPTimer 2B	B	Software	B	GPIO B	B	Software	B
6	Software	B	GPTimer 2A	B	UART5 RX	SB	GPIO C	B	Software	B
7	Software	B	GPTimer 2B	B	UART5 TX	SB	GPIO D	B	Software	B
8	UART0 RX	SB	UART1 RX	SB	Software	B	GPTimer 5A	B	Software	B
9	UART0 TX	SB	UART1 TX	SB	Software	B	GPTimer 5B	B	Software	B
10	SSI0 RX	SB	SSI1 RX	SB	UART6 RX	SB	GPWideTimer 0A	B	Software	B
11	SSI0 TX	SB	SSI1 TX	SB	UART6 TX	SB	GPWideTimer 0B	B	Software	B
12	Software	B	UART2 RX	SB	SSI2 RX	SB	GPWideTimer 1A	B	Software	B
13	Software	B	UART2 TX	SB	SSI2 TX	SB	GPWideTimer 1B	B	Software	B
14	ADC0 SS0	B	GPTimer 2A	B	SSI3 RX	SB	GPIO E	B	Software	B
15	ADC0 SS1	B	GPTimer 2B	B	SSI3 TX	SB	GPIO F	B	Software	B
16	ADC0 SS2	B	Software	B	UART3 RX	SB	GPWideTimer 2A	B	Software	B
17	ADC0 SS3	B	Software	B	UART3 TX	SB	GPWideTimer 2B	B	Software	B
18	GPTimer 0A	B	GPTimer 1A	B	UART4 RX	SB	GPIO B	B	Software	B
19	GPTimer 0B	B	GPTimer 1B	B	UART4 TX	SB	GPIO G	B	Software	B
20	GPTimer 1A	B	Software	B	UART7 RX	SB	Software	B	Software	B

Table 8-1. μDMA Channel Assignments *(continued)*

Enc.	0		1		2		3		4	
Ch #	**Peripheral**	**Type**	**Peripheral**	**Type**	**Peripheral**	**Type**	**Peripheral**	**Type**	**Peripheral**	**Type**
21	GPTimer 1B	B	Software	B	UART7 TX	SB	Software	B	Software	B
22	UART1 RX	SB	Software	B	Software	B	Software	B	Software	B
23	UART1 TX	SB	Software	B	Software	B	Software	B	Software	B
24	SSI1 RX	SB	ADC1 SS0	B	Software	B	GPWideTimer 3A	B	Software	B
25	SSI1 TX	SB	ADC1 SS1	B	Software	B	GPWideTimer 3B	B	Software	B
26	Software	B	ADC1 SS2	B	Software	B	GPWideTimer 4A	B	Software	B
27	Software	B	ADC1 SS3	B	Software	B	GPWideTimer 4B	B	Software	B
28	Software	B	Software	B	Software	B	GPWideTimer 5A	B	Software	B
29	Software	B	Software	B	Software	B	GPWideTimer 5B	B	Software	B
30	Software	B	Software	B	Software	B	Software	B	Software	B
31	Reserved	B	Reserved	B	Reserved	B	Reserved	B	Reserved	B

8.2.2 Priority

The μDMA controller assigns priority to each channel based on the channel number and the priority level bit for the channel. Channel number 0 has the highest priority and as the channel number increases, the priority of a channel decreases. Each channel has a priority level bit to provide two levels of priority: default priority and high priority. If the priority level bit is set, then that channel has higher priority than all other channels at default priority. If multiple channels are set for high priority, then the channel number is used to determine relative priority among all the high priority channels.

The priority bit for a channel can be set using the **DMA Channel Priority Set (DMAPRIOSET)** register and cleared with the **DMA Channel Priority Clear (DMAPRIOCLR)** register.

8.2.3 Arbitration Size

When a μDMA channel requests a transfer, the μDMA controller arbitrates among all the channels making a request and services the μDMA channel with the highest priority. Once a transfer begins, it continues for a selectable number of transfers before rearbitrating among the requesting channels again. The arbitration size can be configured for each channel, ranging from 1 to 1024 item transfers. After the μDMA controller transfers the number of items specified by the arbitration size, it then checks among all the channels making a request and services the channel with the highest priority.

If a lower priority μDMA channel uses a large arbitration size, the latency for higher priority channels is increased because the μDMA controller completes the lower priority burst before checking for higher priority requests. Therefore, lower priority channels should not use a large arbitration size for best response on high priority channels.

The arbitration size can also be thought of as a burst size. It is the maximum number of items that are transferred at any one time in a burst. Here, the term arbitration refers to determination of μDMA channel priority, not arbitration for the bus. When the μDMA controller arbitrates for the bus, the processor always takes priority. Furthermore, the μDMA controller is held off whenever the processor must perform a bus transaction on the same bus, even in the middle of a burst transfer.

8.2.4 Request Types

The μDMA controller responds to two types of requests from a peripheral: single or burst. Each peripheral may support either or both types of requests. A single request means that the peripheral is ready to transfer one item, while a burst request means that the peripheral is ready to transfer multiple items.

The µDMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both are asserted, and the µDMA channel has been set up for a burst transfer, then the burst request takes precedence. See Table 8-2 on page 521, which shows how each peripheral supports the two request types.

Table 8-2. Request Type Support

Peripheral	Event that generates Single Request	Event that generates Burst Request
ADC	None	FIFO half full
General-Purpose Timer	None	Trigger event
GPIO	Raw interrupt pulse	None
SSI TX	TX FIFO Not Full	TX FIFO Level (fixed at 4)
SSI RX	RX FIFO Not Empty	RX FIFO Level (fixed at 4)
UART TX	TX FIFO Not Full	TX FIFO Level (configurable)
UART RX	RX FIFO Not Empty	RX FIFO Level (configurable)
USB TX	None	FIFO TXRDY
USB RX	None	FIFO RXRDY

8.2.4.1 Single Request

When a single request is detected, and not a burst request, the µDMA controller transfers one item and then stops to wait for another request.

8.2.4.2 Burst Request

When a burst request is detected, the µDMA controller transfers the number of items that is the lesser of the arbitration size or the number of items remaining in the transfer. Therefore, the arbitration size should be the same as the number of data items that the peripheral can accommodate when making a burst request. For example, the UART generates a burst request based on the FIFO trigger level. In this case, the arbitration size should be set to the amount of data that the FIFO can transfer when the trigger level is reached. A burst transfer runs to completion once it is started, and cannot be interrupted, even by a higher priority channel. Burst transfers complete in a shorter time than the same number of non-burst transfers.

It may be desirable to use only burst transfers and not allow single transfers. For example, perhaps the nature of the data is such that it only makes sense when transferred together as a single unit rather than one piece at a time. The single request can be disabled by using the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register. By setting the bit for a channel in this register, the µDMA controller only responds to burst requests for that channel.

8.2.5 Channel Configuration

The µDMA controller uses an area of system memory to store a set of channel control structures in a table. The control table may have one or two entries for each µDMA channel. Each entry in the table structure contains source and destination pointers, transfer size, and transfer mode. The control table can be located anywhere in system memory, but it must be contiguous and aligned on a 1024-byte boundary.

Table 8-3 on page 522 shows the layout in memory of the channel control table. Each channel may have one or two control structures in the control table: a primary control structure and an optional alternate control structure. The table is organized so that all of the primary entries are in the first half of the table, and all the alternate structures are in the second half of the table. The primary entry is used for simple transfer modes where transfers can be reconfigured and restarted after each transfer is complete. In this case, the alternate control structures are not used and therefore only the first half of the table must be allocated in memory; the second half of the control table is not necessary, and that memory can be used for something else. If a more complex transfer mode is used such as ping-pong or scatter-gather, then the alternate control structure is also used and memory space should be allocated for the entire table.

Any unused memory in the control table may be used by the application. This includes the control structures for any channels that are unused by the application as well as the unused control word for each channel.

Table 8-3. Control Structure Memory Map

Offset	Channel
0x0	0, Primary
0x10	1, Primary
...	...
0x1F0	31, Primary
0x200	0, Alternate
0x210	1, Alternate
...	...
0x3F0	31, Alternate

Table 8-4 shows an individual control structure entry in the control table. Each entry is aligned on a 16-byte boundary. The entry contains four long words: the source end pointer, the destination end pointer, the control word, and an unused entry. The end pointers point to the ending address of the transfer and are inclusive. If the source or destination is non-incrementing (as for a peripheral register), then the pointer should point to the transfer address.

Table 8-4. Channel Control Structure

Offset	Description
0x000	Source End Pointer
0x004	Destination End Pointer
0x008	Control Word
0x00C	Unused

The control word contains the following fields:

- Source and destination data sizes
- Source and destination address increment size
- Number of transfers before bus arbitration
- Total number of items to transfer
- Useburst flag
- Transfer mode

The control word and each field are described in detail in “μDMA Channel Control Structure” on page 540. The μDMA controller updates the transfer size and transfer mode fields as the transfer is performed. At the end of a transfer, the transfer size indicates 0, and the transfer mode indicates "stopped." Because the control word is modified by the μDMA controller, it must be reconfigured before each new transfer. The source and destination end pointers are not modified, so they can be left unchanged if the source or destination addresses remain the same.

Prior to starting a transfer, a μDMA channel must be enabled by setting the appropriate bit in the **DMA Channel Enable Set (DMAENASET)** register. A channel can be disabled by setting the channel bit in the **DMA Channel Enable Clear (DMAENACLR)** register. At the end of a complete μDMA transfer, the controller automatically disables the channel.

8.2.6 Transfer Modes

The μDMA controller supports several transfer modes. Two of the modes support simple one-time transfers. Several complex modes support a continuous flow of data.

8.2.6.1 Stop Mode

While Stop is not actually a transfer mode, it is a valid value for the mode field of the control word. When the mode field has this value, the μDMA controller does not perform any transfers and disables the channel if it is enabled. At the end of a transfer, the μDMA controller updates the control word to set the mode to Stop.

8.2.6.2 Basic Mode

In Basic mode, the μDMA controller performs transfers as long as there are more items to transfer, and a transfer request is present. This mode is used with peripherals that assert a μDMA request signal whenever the peripheral is ready for a data transfer. Basic mode should not be used in any situation where the request is momentary even though the entire transfer should be completed. For example, a software-initiated transfer creates a momentary request, and in Basic mode, only the number of transfers specified by the ARBSIZE field in the **DMA Channel Control Word (DMACHCTL)** register is transferred on a software request, even if there is more data to transfer.

When all of the items have been transferred using Basic mode, the μDMA controller sets the mode for that channel to Stop.

8.2.6.3 Auto Mode

Auto mode is similar to Basic mode, except that once a transfer request is received, the transfer runs to completion, even if the μDMA request is removed. This mode is suitable for software-triggered transfers. Generally, Auto mode is not used with a peripheral.

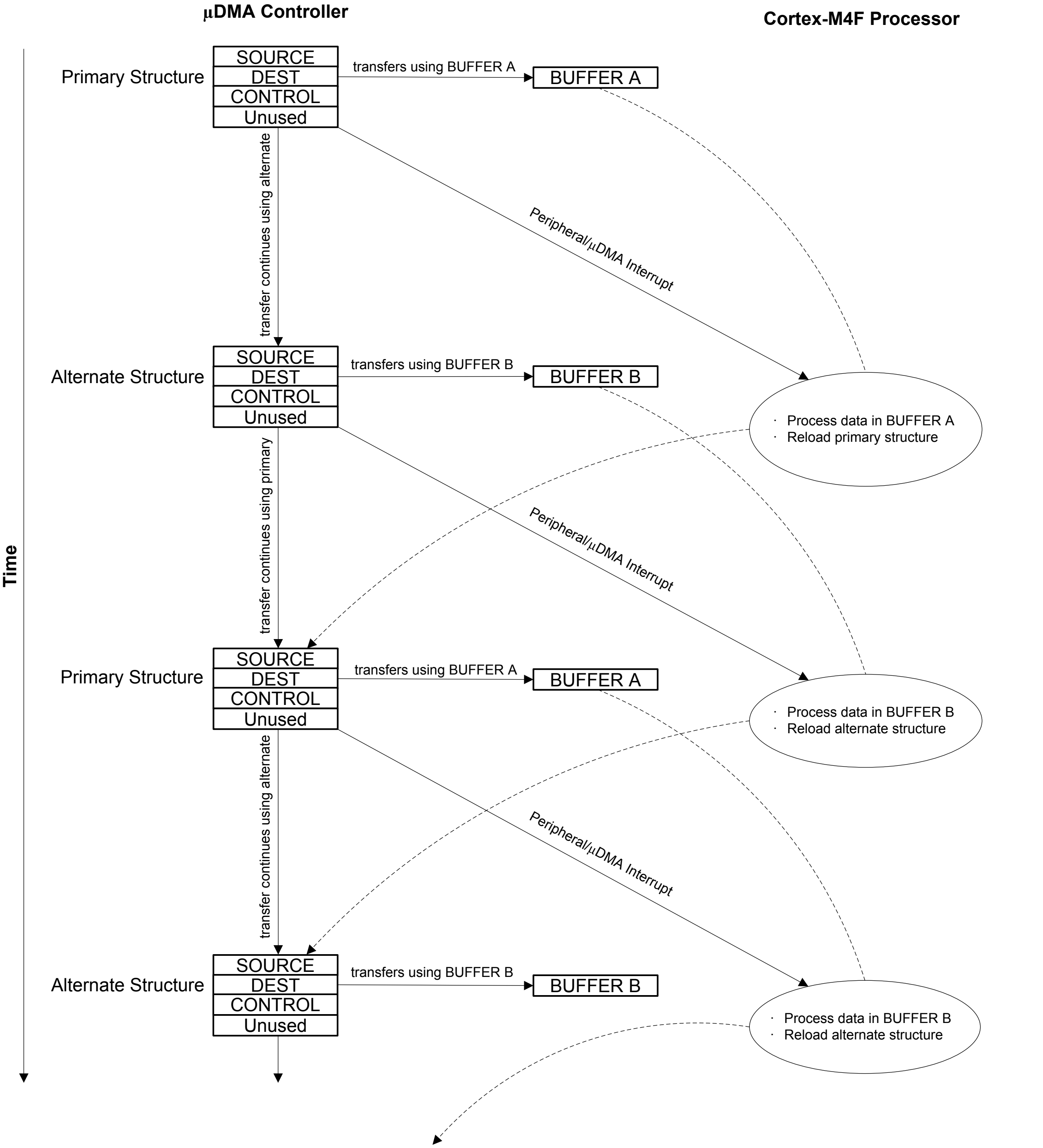
When all the items have been transferred using Auto mode, the μDMA controller sets the mode for that channel to Stop.

8.2.6.4 Ping-Pong

Ping-Pong mode is used to support a continuous data flow to or from a peripheral. To use Ping-Pong mode, both the primary and alternate data structures must be implemented. Both structures are set up by the processor for data transfer between memory and a peripheral. The transfer is started using the primary control structure. When the transfer using the primary control structure is complete, the μDMA controller reads the alternate control structure for that channel to continue the transfer. Each time this happens, an interrupt is generated, and the processor can reload the control structure for the just-completed transfer. Data flow can continue indefinitely this way, using the primary and alternate control structures to switch back and forth between buffers as the data flows to or from the peripheral.

Refer to Figure 8-2 on page 524 for an example showing operation in Ping-Pong mode.

Figure 8-2. Example of Ping-Pong μDMA Transaction


μDMA Controller
Cortex-M4F Processor
Primary Structure
SOURCE
DEST
CONTROL
Unused
transfers using BUFFER A
BUFFER A
transfer continues using alternate
Peripheral/μDMA Interrupt
Alternate Structure
SOURCE
DEST
CONTROL
Unused
transfers using BUFFER B
BUFFER B
· Process data in BUFFER A
· Reload primary structure
transfer continues using primary
Peripheral/μDMA Interrupt
Time
Primary Structure
SOURCE
DEST
CONTROL
Unused
transfers using BUFFER A
BUFFER A
· Process data in BUFFER B
· Reload alternate structure
transfer continues using alternate
Peripheral/μDMA Interrupt
Alternate Structure
SOURCE
DEST
CONTROL
Unused
transfers using BUFFER B
BUFFER B
· Process data in BUFFER B
· Reload alternate structure

8.2.6.5 Memory Scatter-Gather

Memory Scatter-Gather mode is a complex mode used when data must be transferred to or from varied locations in memory instead of a set of contiguous locations in a memory buffer. For example, a gather μDMA operation could be used to selectively read the payload of several stored packets of a communication protocol and store them together in sequence in a memory buffer.

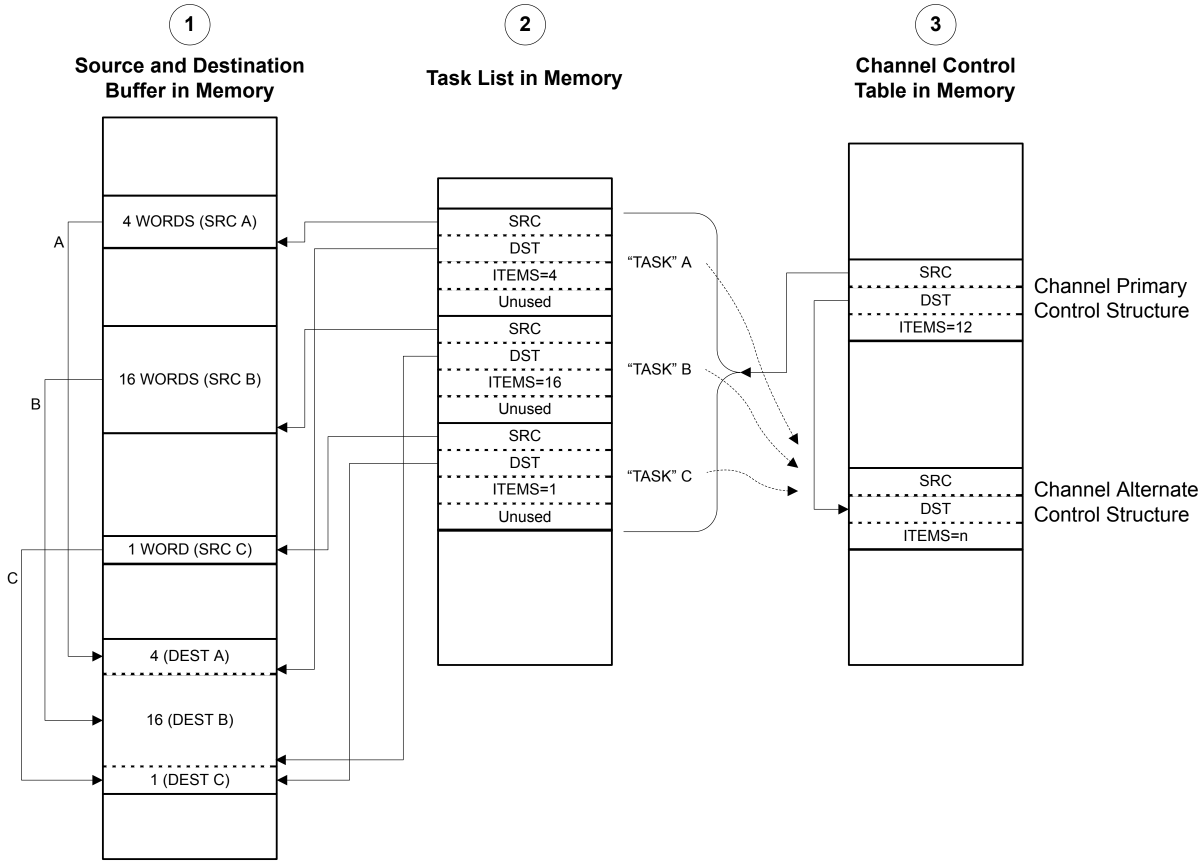
In Memory Scatter-Gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word for a specific transfer. The mode of each control word must be set to Scatter-Gather mode. Each entry in the table is copied in turn to the alternate structure where it is then executed. The μDMA controller alternates between using the primary control structure to copy the next transfer instruction from the list and then executing the new transfer instruction. The end of the list is marked by programming the control word for the last entry to use Auto transfer mode. Once the last transfer is performed using Auto mode, the μDMA controller stops. A completion interrupt is generated only after the last transfer. It is possible to loop the list by having the last entry copy the primary control structure to point back to the beginning of the list (or to a new list). It is also possible to trigger a set of other channels to perform a transfer, either directly, by programming a write to the software trigger for another channel, or indirectly, by causing a peripheral action that results in a μDMA request.

By programming the μDMA controller using this method, a set of up to 256 arbitrary transfers can be performed based on a single μDMA request.

Refer to Figure 8-3 on page 526 and Figure 8-4 on page 527, which show an example of operation in Memory Scatter-Gather mode. This example shows a *gather* operation, where data in three separate buffers in memory is copied together into one buffer. Figure 8-3 on page 526 shows how the application sets up a μDMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

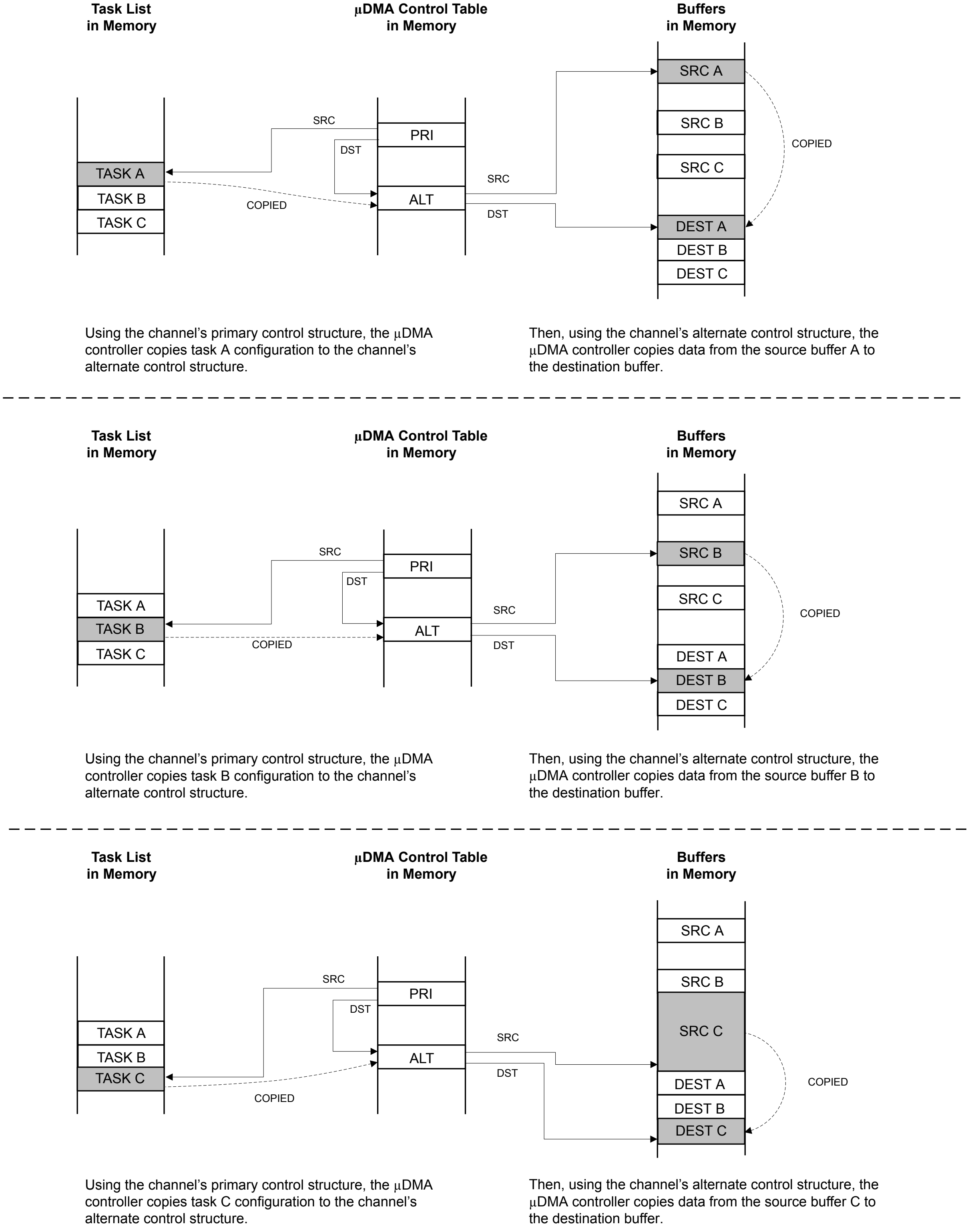
Figure 8-4 on page 527 shows the sequence as the μDMA controller performs the three sets of copy operations. First, using the primary control structure, the μDMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the destination buffer. Next, the μDMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

Figure 8-3. Memory Scatter-Gather, Setup and Configuration



NOTES:
1. Application has a need to copy data items from three separate locations in memory into one combined buffer.
2. Application sets up µDMA "task list" in memory, which contains the pointers and control configuration for three µDMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the µDMA controller.
4. The SRC and DST pointers in the task list must point to the last location in the corresponding buffer.

Figure 8-4. Memory Scatter-Gather, µDMA Copy Sequence



Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer A to the destination buffer.

Using the channel's primary control structure, the µDMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer B to the destination buffer.

Using the channel's primary control structure, the µDMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer C to the destination buffer.

8.2.6.6 Peripheral Scatter-Gather

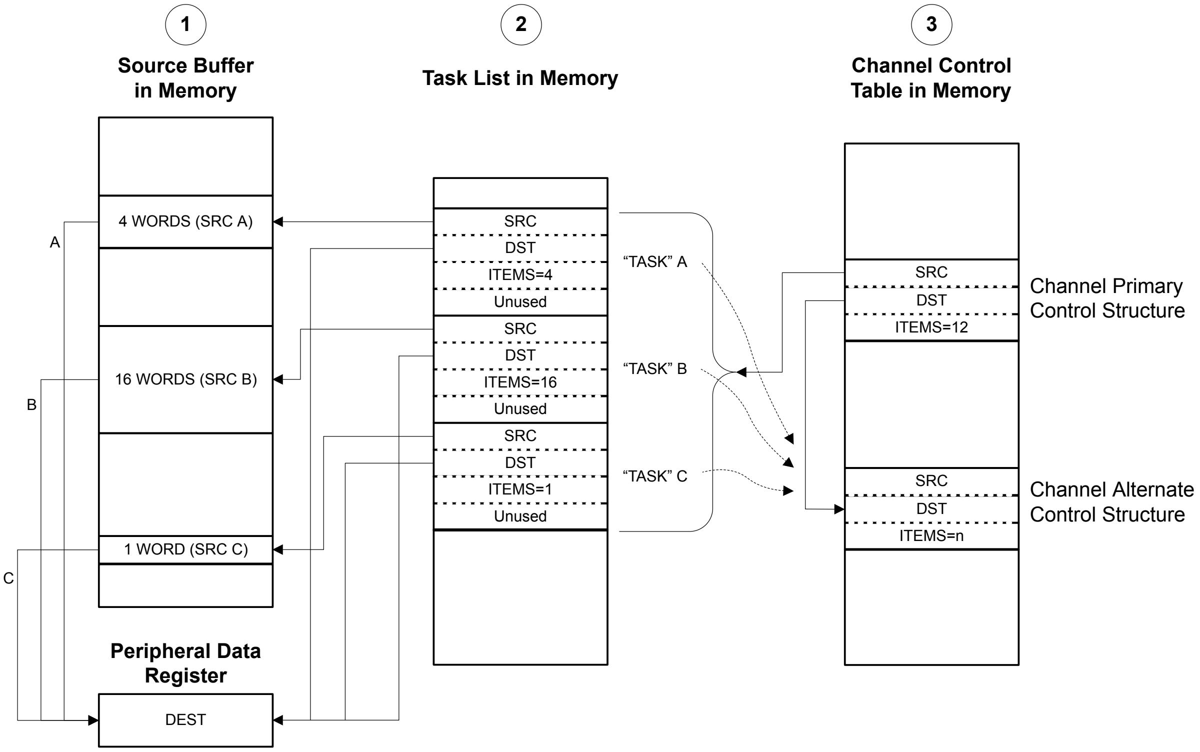
Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a μDMA request. Upon detecting a request from the peripheral, the μDMA controller uses the primary control structure to copy one entry from the list to the alternate control structure and then performs the transfer. At the end of this transfer, the next transfer is started only if the peripheral again asserts a μDMA request. The μDMA controller continues to perform transfers from the list only when the peripheral is making a request, until the last transfer is complete. A completion interrupt is generated only after the last transfer.

By using this method, the μDMA controller can transfer data to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

Refer to Figure 8-5 on page 529 and Figure 8-6 on page 530, which show an example of operation in Peripheral Scatter-Gather mode. This example shows a gather operation, where data from three separate buffers in memory is copied to a single peripheral data register. Figure 8-5 on page 529 shows how the application sets up a μDMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

Figure 8-6 on page 530 shows the sequence as the μDMA controller performs the three sets of copy operations. First, using the primary control structure, the μDMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the peripheral data register. Next, the μDMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

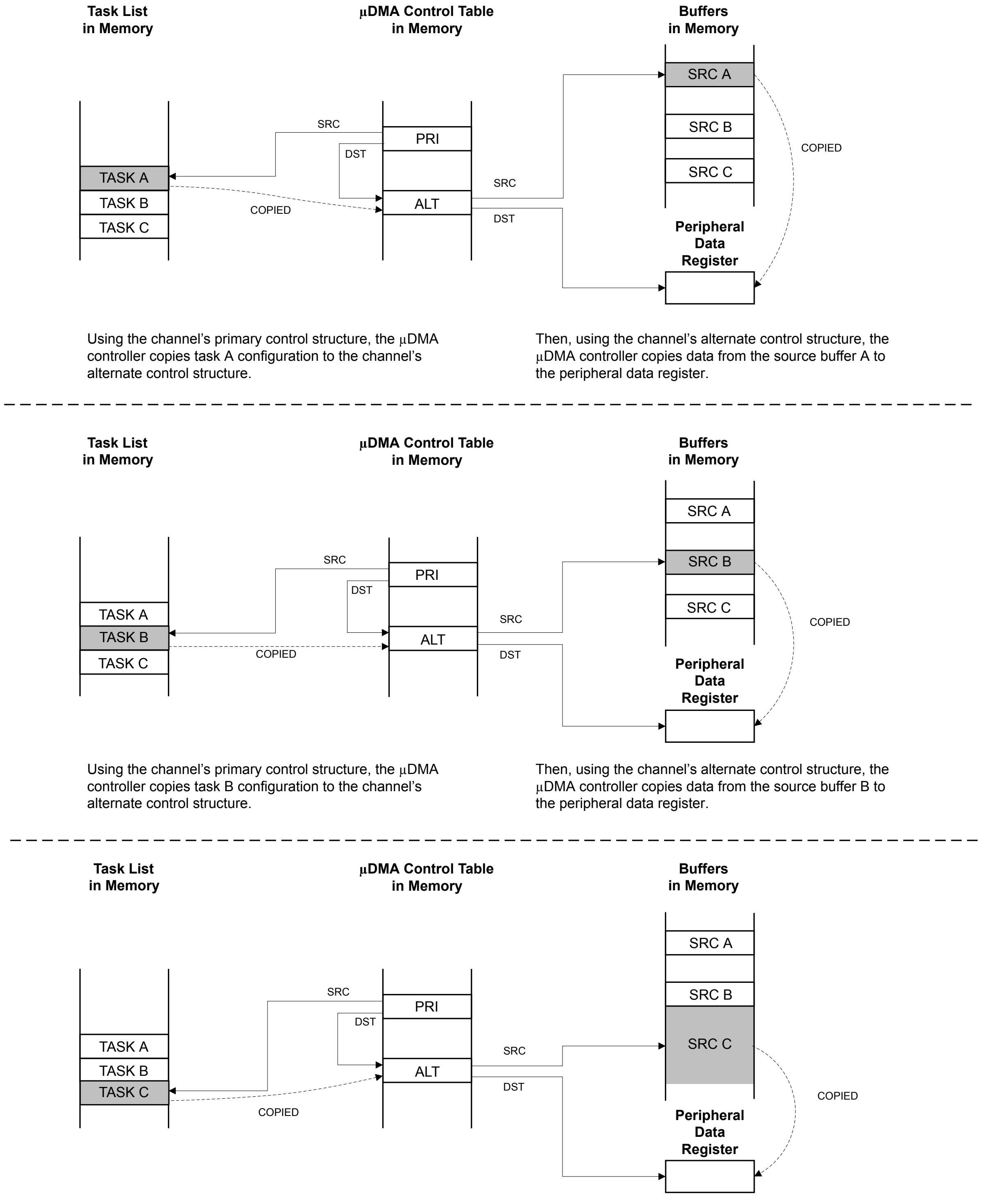
Figure 8-5. Peripheral Scatter-Gather, Setup and Configuration



NOTES:

1. Application has a need to copy data items from three separate locations in memory into a peripheral data register.
2. Application sets up μDMA "task list" in memory, which contains the pointers and control configuration for three μDMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the μDMA controller.

Figure 8-6. Peripheral Scatter-Gather, μDMA Copy Sequence



Using the channel's primary control structure, the μDMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μDMA controller copies data from the source buffer A to the peripheral data register.

Using the channel's primary control structure, the μDMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μDMA controller copies data from the source buffer B to the peripheral data register.

Using the channel's primary control structure, the μDMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μDMA controller copies data from the source buffer C to the peripheral data register.

8.2.7 Transfer Size and Increment

The μDMA controller supports transfer data sizes of 8, 16, or 32 bits. The source and destination data size must be the same for any given transfer. The source and destination address can be auto-incremented by bytes, half-words, or words, or can be set to no increment. The source and destination address increment values can be set independently, and it is not necessary for the address increment to match the data size as long as the increment is the same or larger than the data size. For example, it is possible to perform a transfer using 8-bit data size, but using an address increment of full words (4 bytes). The data to be transferred must be aligned in memory according to the data size (8, 16, or 32 bits).

Table 8-5 shows the configuration to read from a peripheral that supplies 8-bit data.

Table 8-5. μDMA Read Example: 8-Bit Peripheral

Field	Configuration
Source data size	8 bits
Destination data size	8 bits
Source address increment	No increment
Destination address increment	Byte
Source end pointer	Peripheral read FIFO register
Destination end pointer	End of the data buffer in memory

8.2.8 Peripheral Interface

Each peripheral that supports μDMA has a single request and/or burst request signal that is asserted when the peripheral is ready to transfer data (see Table 8-2 on page 521). The request signal can be disabled or enabled using the **DMA Channel Request Mask Set (DMAREQMASKSET)** and **DMA Channel Request Mask Clear (DMAREQMASKCLR)** registers. The μDMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the μDMA channel is configured correctly and enabled, and the peripheral asserts the request signal, the μDMA controller begins the transfer.

Note: When using μDMA to transfer data to and from a peripheral, the peripheral must disable all interrupts to the NVIC.

When a μDMA transfer is complete, the μDMA controller generates an interrupt, see “Interrupts and Errors” on page 532 for more information.

For more information on how a specific peripheral interacts with the μDMA controller, refer to the DMA Operation section in the chapter that discusses that peripheral.

8.2.9 Software Request

One μDMA channel is dedicated to software-initiated transfers. This channel also has a dedicated interrupt to signal completion of a μDMA transfer. A transfer is initiated by software by first configuring and enabling the transfer, and then issuing a software request using the **DMA Channel Software Request (DMASWREQ)** register. For software-based transfers, the Auto transfer mode should be used.

It is possible to initiate a transfer on any available software channel using the **DMASWREQ** register. If a request is initiated by software using a peripheral μDMA channel, then the completion interrupt occurs on the interrupt vector for the peripheral instead of the software interrupt vector. Any peripheral channel may be used for software requests as long as the corresponding peripheral is not using μDMA for data transfer.

8.2.10 Interrupts and Errors

Depending on the peripheral, the μDMA can indicate transfer completion at the end of an entire transfer or when a FIFO or buffer reaches a certain level (see Table 8-2 on page 521 and the individual peripheral chapters). When a μDMA transfer is complete, the μDMA controller generates a completion interrupt on the interrupt vector of the peripheral. Therefore, if μDMA is used to transfer data for a peripheral and interrupts are used, then the interrupt handler for that peripheral must be designed to handle the μDMA transfer completion interrupt. If the transfer uses the software μDMA channel, then the completion interrupt occurs on the dedicated software μDMA interrupt vector (see Table 8-6 on page 532).

When μDMA is enabled for a peripheral, the μDMA controller stops the normal transfer interrupts for a peripheral from reaching the interrupt controller (the interrupts are still reported in the peripheral's interrupt registers). Thus, when a large amount of data is transferred using μDMA, instead of receiving multiple interrupts from the peripheral as data flows, the interrupt controller receives only one interrupt when the transfer is complete. Unmasked peripheral error interrupts continue to be sent to the interrupt controller.

When a μDMA channel generates a completion interrupt, the CHIS bit corresponding to the peripheral channel is set in the **DMA Channel Interrupt Status (DMACHIS)** register (see page 567). This register can be used by the peripheral interrupt handler code to determine if the interrupt was caused by the μDMA channel or an error event reported by the peripheral's interrupt registers. The completion interrupt request from the μDMA controller is automatically cleared when the interrupt handler is activated.

When transfers are performed from a FIFO of the UART or SSI using the μDMA, and any interrupt is generated from the UART or SSI, the module's status bit in the **DMA Channel Interrupt Status (DMACHIS)** register must be checked at the end of the interrupt service routine. If the status bit is set, clear the interrupt by writing a 1 to it.

If the μDMA controller encounters a bus or memory protection error as it attempts to perform a data transfer, it disables the μDMA channel that caused the error and generates an interrupt on the μDMA error interrupt vector. The processor can read the **DMA Bus Error Clear (DMAERRCLR)** register to determine if an error is pending. The ERRCLR bit is set if an error occurred. The error can be cleared by writing a 1 to the ERRCLR bit.

Table 8-6 shows the dedicated interrupt assignments for the μDMA controller.

Table 8-6. μDMA Interrupt Assignments

Interrupt	Assignment
46	μDMA Software Channel Transfer
47	μDMA Error

8.3 Initialization and Configuration

8.3.1 Module Initialization

Before the μDMA controller can be used, it must be enabled in the System Control block and in the peripheral. The location of the channel control structure must also be programmed.

The following steps should be performed one time during system initialization:

1. Enable the μDMA clock using the **RCGCDMA** register (see page 321).

2. Enable the μDMA controller by setting the MASTEREN bit of the **DMA Configuration (DMACFG)** register.

3. Program the location of the channel control table by writing the base address of the table to the **DMA Channel Control Base Pointer (DMACTLBASE)** register. The base address must be aligned on a 1024-byte boundary.

8.3.2 Configuring a Memory-to-Memory Transfer

μDMA channel 30 is dedicated for software-initiated transfers. However, any channel can be used for software-initiated, memory-to-memory transfer if the associated peripheral is not being used.

8.3.2.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Program bit 30 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.

2. Set bit 30 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.

3. Set bit 30 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the μDMA controller to respond to single and burst requests.

4. Set bit 30 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the μDMA controller to recognize requests for this channel.

8.3.2.2 Configure the Channel Control Structure

Now the channel control structure must be configured.

This example transfers 256 words from one memory buffer to another. Channel 30 is used for a software transfer, and the control structure for channel 30 is at offset 0x1E0 of the channel control table. The channel control structure for channel 30 is located at the offsets shown in Table 8-7.

Table 8-7. Channel Control Structure Offsets for Channel 30

Offset	Description
Control Table Base + 0x1E0	Channel 30 Source End Pointer
Control Table Base + 0x1E4	Channel 30 Destination End Pointer
Control Table Base + 0x1E8	Channel 30 Control Word

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive).

1. Program the source end pointer at offset 0x1E0 to the address of the source buffer + 0x3FC.

2. Program the destination end pointer at offset 0x1E4 to the address of the destination buffer + 0x3FC.

The control word at offset 0x1E8 must be programmed according to Table 8-8.

Table 8-8. Channel Control Word Configuration for Memory Transfer Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	2	32-bit destination address increment
DSTSIZE	29:28	2	32-bit destination data size
SRCINC	27:26	2	32-bit source address increment
SRCSIZE	25:24	2	32-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	255	Transfer 256 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	2	Use Auto-request transfer mode

8.3.2.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 30 of the **DMA Channel Enable Set (DMAENASET)** register.
2. Issue a transfer request by setting bit 30 of the **DMA Channel Software Request (DMASWREQ)** register.

The μDMA transfer begins. If the interrupt is enabled, then the processor is notified by interrupt when the transfer is complete. If needed, the status can be checked by reading bit 30 of the **DMAENASET** register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the XFERMODE field of the channel control word at offset 0x1E8. This field is automatically cleared at the end of the transfer.

8.3.3 Configuring a Peripheral for Simple Transmit

This example configures the μDMA controller to transmit a buffer of data to a peripheral. The peripheral has a transmit FIFO with a trigger level of 4. The example peripheral uses μDMA channel 7.

8.3.3.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 7 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 7 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 7 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the μDMA controller to respond to single and burst requests.
4. Set bit 7 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the μDMA controller to recognize requests for this channel.

8.3.3.2 Configure the Channel Control Structure

This example transfers 64 bytes from a memory buffer to the peripheral's transmit FIFO register using μDMA channel 7. The control structure for channel 7 is at offset 0x070 of the channel control table. The channel control structure for channel 7 is located at the offsets shown in Table 8-9.

Table 8-9. Channel Control Structure Offsets for Channel 7

Offset	Description
Control Table Base + 0x070	Channel 7 Source End Pointer
Control Table Base + 0x074	Channel 7 Destination End Pointer
Control Table Base + 0x078	Channel 7 Control Word

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register.

1. Program the source end pointer at offset 0x070 to the address of the source buffer + 0x3F.
2. Program the destination end pointer at offset 0x074 to the address of the peripheral's transmit FIFO register.

The control word at offset 0x078 must be programmed according to Table 8-10.

Table 8-10. Channel Control Word Configuration for Peripheral Transmit Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	3	Destination address does not increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	0	8-bit source address increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	2	Arbitrates after 4 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	1	Use Basic transfer mode

Note: In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 4, the arbitration size is set to 4. If the peripheral does make a burst request, then 4 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any space in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst SET[7] bit should be set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

8.3.3.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 7 of the **DMA Channel Enable Set (DMAENASET)** register.

The μDMA controller is now configured for transfer on channel 7. The controller makes transfers to the peripheral whenever the peripheral asserts a μDMA request. The transfers continue until the entire buffer of 64 bytes has been transferred. When that happens, the μDMA controller disables the channel and sets the `XFERMODE` field of the channel control word to 0 (Stopped). The status of the transfer can be checked by reading bit 7 of the **DMA Channel Enable Set (DMAENASET)** register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the `XFERMODE` field of the channel control word at offset 0x078. This field is automatically cleared at the end of the transfer.

If peripheral interrupts are enabled, then the peripheral interrupt handler receives an interrupt when the entire transfer is complete.

8.3.4 Configuring a Peripheral for Ping-Pong Receive

This example configures the μDMA controller to continuously receive 8-bit data from a peripheral into a pair of 64-byte buffers. The peripheral has a receive FIFO with a trigger level of 8. The example peripheral uses μDMA channel 8.

8.3.4.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 8 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 8 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 8 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the μDMA controller to respond to single and burst requests.
4. Set bit 8 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the μDMA controller to recognize requests for this channel.

8.3.4.2 Configure the Channel Control Structure

This example transfers bytes from the peripheral's receive FIFO register into two memory buffers of 64 bytes each. As data is received, when one buffer is full, the μDMA controller switches to use the other.

To use Ping-Pong buffering, both primary and alternate channel control structures must be used. The primary control structure for channel 8 is at offset 0x080 of the channel control table, and the alternate channel control structure is at offset 0x280. The channel control structures for channel 8 are located at the offsets shown in Table 8-11.

Table 8-11. Primary and Alternate Channel Control Structure Offsets for Channel 8

Offset	Description
Control Table Base + 0x080	Channel 8 Primary Source End Pointer
Control Table Base + 0x084	Channel 8 Primary Destination End Pointer
Control Table Base + 0x088	Channel 8 Primary Control Word
Control Table Base + 0x280	Channel 8 Alternate Source End Pointer
Control Table Base + 0x284	Channel 8 Alternate Destination End Pointer
Control Table Base + 0x288	Channel 8 Alternate Control Word

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register. Both the primary and alternate sets of pointers must be configured.

1. Program the primary source end pointer at offset 0x080 to the address of the peripheral's receive buffer.
2. Program the primary destination end pointer at offset 0x084 to the address of ping-pong buffer A + 0x3F.
3. Program the alternate source end pointer at offset 0x280 to the address of the peripheral's receive buffer.
4. Program the alternate destination end pointer at offset 0x284 to the address of ping-pong buffer B + 0x3F.

The primary control word at offset 0x088 and the alternate control word at offset 0x288 are initially programmed the same way.

1. Program the primary channel control word at offset 0x088 according to Table 8-12.
2. Program the alternate channel control word at offset 0x288 according to Table 8-12.

Table 8-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	0	8-bit destination address increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	3	Source address does not increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	3	Use Ping-Pong transfer mode

Note: In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 8, the arbitration size is set to 8. If the peripheral does make a burst request, then 8 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any data in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst `SET[8]` bit should be set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

8.3.4.3 Configure the Peripheral Interrupt

An interrupt handler should be configured when using µDMA Ping-Pong mode, it is best to use an interrupt handler. However, the Ping-Pong mode can be configured without interrupts by polling. The interrupt handler is triggered after each buffer is complete.

1. Configure and enable an interrupt handler for the peripheral.

8.3.4.4 Enable the μDMA Channel

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 8 of the **DMA Channel Enable Set (DMAENASET)** register.

8.3.4.5 Process Interrupts

The μDMA controller is now configured and enabled for transfer on channel 8. When the peripheral asserts the μDMA request signal, the μDMA controller makes transfers into buffer A using the primary channel control structure. When the primary transfer to buffer A is complete, it switches to the alternate channel control structure and makes transfers into buffer B. At the same time, the primary channel control word mode field is configured to indicate Stopped, and an interrupt is pending.

When an interrupt is triggered, the interrupt handler must determine which buffer is complete and process the data or set a flag that the data must be processed by non-interrupt buffer processing code. Then the next buffer transfer must be set up.

In the interrupt handler:

1. Read the primary channel control word at offset 0x088 and check the XFERMODE field. If the field is 0, this means buffer A is complete. If buffer A is complete, then:
   a. Process the newly received data in buffer A or signal the buffer processing code that buffer A has data available.
   b. Reprogram the primary channel control word at offset 0x88 according to Table 8-12 on page 537.
2. Read the alternate channel control word at offset 0x288 and check the XFERMODE field. If the field is 0, this means buffer B is complete. If buffer B is complete, then:
   a. Process the newly received data in buffer B or signal the buffer processing code that buffer B has data available.
   b. Reprogram the alternate channel control word at offset 0x288 according to Table 8-12 on page 537.

8.3.5 Configuring Channel Assignments

Channel assignments for each μDMA channel can be changed using the **DMACHMAPn** registers. Each 4-bit field represents a μDMA channel.

Refer to Table 8-1 on page 519 for channel assignments.

8.4 Register Map

Table 8-13 on page 539 lists the μDMA channel control structures and registers. The channel control structure shows the layout of one entry in the channel control table. The channel control table is located in system memory, and the location is determined by the application, thus the base address is n/a (not applicable) and noted as such above the register descriptions. In the table below, the offset for the channel control structures is the offset from the entry in the channel control table. See "Channel Configuration" on page 521 and Table 8-3 on page 522 for a description of how the entries in the channel control table are located in memory. The μDMA register addresses are given as a hexadecimal increment, relative to the μDMA base address of 0x400F.F000. Note that the μDMA module clock must be enabled before the registers can be programmed (see page 321). There must be a delay of 3 system clocks after the µDMA module clock is enabled before any µDMA module registers are accessed.

Table 8-13. µDMA Register Map

Offset	Name	Type	Reset	Description	See page
µDMA Channel Control Structure (Offset from Channel Control Table Base)					
0x000	DMASRCENDP	RW	-	DMA Channel Source Address End Pointer	541
0x004	DMADSTENDP	RW	-	DMA Channel Destination Address End Pointer	542
0x008	DMACHCTL	RW	-	DMA Channel Control Word	543
µDMA Registers (Offset from µDMA Base Address)					
0x000	DMASTAT	RO	0x001F.0000	DMA Status	548
0x004	DMACFG	WO	-	DMA Configuration	550
0x008	DMACTLBASE	RW	0x0000.0000	DMA Channel Control Base Pointer	551
0x00C	DMAALTBASE	RO	0x0000.0200	DMA Alternate Channel Control Base Pointer	552
0x010	DMAWAITSTAT	RO	0x03C3.CF00	DMA Channel Wait-on-Request Status	553
0x014	DMASWREQ	WO	-	DMA Channel Software Request	554
0x018	DMAUSEBURSTSET	RW	0x0000.0000	DMA Channel Useburst Set	555
0x01C	DMAUSEBURSTCLR	WO	-	DMA Channel Useburst Clear	556
0x020	DMAREQMASKSET	RW	0x0000.0000	DMA Channel Request Mask Set	557
0x024	DMAREQMASKCLR	WO	-	DMA Channel Request Mask Clear	558
0x028	DMAENASET	RW	0x0000.0000	DMA Channel Enable Set	559
0x02C	DMAENACLR	WO	-	DMA Channel Enable Clear	560
0x030	DMAALTSET	RW	0x0000.0000	DMA Channel Primary Alternate Set	561
0x034	DMAALTCLR	WO	-	DMA Channel Primary Alternate Clear	562
0x038	DMAPRIOSET	RW	0x0000.0000	DMA Channel Priority Set	563
0x03C	DMAPRIOCLR	WO	-	DMA Channel Priority Clear	564
0x04C	DMAERRCLR	RW	0x0000.0000	DMA Bus Error Clear	565
0x500	DMACHASGN	RW	0x0000.0000	DMA Channel Assignment	566
0x504	DMACHIS	RW1C	0x0000.0000	DMA Channel Interrupt Status	567
0x510	DMACHMAP0	RW	0x0000.0000	DMA Channel Map Select 0	568
0x514	DMACHMAP1	RW	0x0000.0000	DMA Channel Map Select 1	569
0x518	DMACHMAP2	RW	0x0000.0000	DMA Channel Map Select 2	570
0x51C	DMACHMAP3	RW	0x0000.0000	DMA Channel Map Select 3	571
0xFD0	DMAPeriphID4	RO	0x0000.0004	DMA Peripheral Identification 4	576
0xFE0	DMAPeriphID0	RO	0x0000.0030	DMA Peripheral Identification 0	572

Table 8-13. μDMA Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFE4	DMAPeriphID1	RO	0x0000.00B2	DMA Peripheral Identification 1	573
0xFE8	DMAPeriphID2	RO	0x0000.000B	DMA Peripheral Identification 2	574
0xFEC	DMAPeriphID3	RO	0x0000.0000	DMA Peripheral Identification 3	575
0xFF0	DMAPCellID0	RO	0x0000.000D	DMA PrimeCell Identification 0	577
0xFF4	DMAPCellID1	RO	0x0000.00F0	DMA PrimeCell Identification 1	578
0xFF8	DMAPCellID2	RO	0x0000.0005	DMA PrimeCell Identification 2	579
0xFFC	DMAPCellID3	RO	0x0000.00B1	DMA PrimeCell Identification 3	580

8.5 μDMA Channel Control Structure

The μDMA Channel Control Structure holds the transfer settings for a μDMA channel. Each channel has two control structures, which are located in a table in system memory. Refer to “Channel Configuration” on page 521 for an explanation of the Channel Control Table and the Channel Control Structure.

The channel control structure is one entry in the channel control table. Each channel has a primary and alternate structure. The primary control structures are located at offsets 0x0, 0x10, 0x20 and so on. The alternate control structures are located at offsets 0x200, 0x210, 0x220, and so on.

Register 1: DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000

DMA Channel Source Address End Pointer (DMASRCENDP) is part of the Channel Control Structure and is used to specify the source address for a μDMA transfer.

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

Note: The offset specified is from the base address of the control structure in system memory, not the μDMA module base address.

DMA Channel Source Address End Pointer (DMASRCENDP)

Base n/a
Offset 0x000
Type RW, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:0	ADDR	RW	-	Source Address End Pointer This field points to the last address of the μDMA transfer source (inclusive). If the source address is not incrementing (the SRCINC field in the **DMACHCTL** register is 0x3), then this field points at the source location itself (such as a peripheral data register).

Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004

DMA Channel Destination Address End Pointer (DMADSTENDP) is part of the Channel Control Structure and is used to specify the destination address for a μDMA transfer.

Note: The offset specified is from the base address of the control structure in system memory, not the μDMA module base address.

DMA Channel Destination Address End Pointer (DMADSTENDP)

Base n/a
Offset 0x004
Type RW, reset -

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

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

Bit/Field	Name	Type	Reset	Description
31:0	ADDR	RW	-	Destination Address End Pointer This field points to the last address of the μDMA transfer destination (inclusive). If the destination address is not incrementing (the `DSTINC` field in the **DMACHCTL** register is 0x3), then this field points at the destination location itself (such as a peripheral data register).

Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008

DMA Channel Control Word (DMACHCTL) is part of the Channel Control Structure and is used to specify parameters of a µDMA transfer.

Note: The offset specified is from the base address of the control structure in system memory, not the µDMA module base address.

DMA Channel Control Word (DMACHCTL)

Base n/a
Offset 0x008
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DSTINC		DSTSIZE		SRCINC		SRCSIZE		reserved						ARBSIZE	
Type	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RO	RO	RO	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ARBSIZE		XFERSIZE										NXTUSEBURST	XFERMODE		
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:30	DSTINC	RW	-	Destination Address Increment This field configures the destination address increment. The address increment value must be equal or greater than the value of the destination size (DSTSIZE).

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	RW	-	Destination Data Size This field configures the destination item data size. **Note:** DSTSIZE must be the same as SRCSIZE. 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	RW	-	Source Address Increment This field configures the source address increment. The address increment value must be equal or greater than the value of the source size (SRCSIZE). 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 (DMASRCENDP) for the channel
25:24	SRCSIZE	RW	-	Source Data Size This field configures the source item data size. **Note:** DSTSIZE must be the same as SRCSIZE. Value Description 0x0 Byte 8-bit data size. 0x1 Half-word 16-bit data size. 0x2 Word 32-bit data size. 0x3 Reserved
23:18	reserved	RO	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
17:14	ARBSIZE	RW	-	Arbitration Size 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. 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 In this configuration, no arbitration occurs during the µDMA transfer because the maximum transfer size is 1024.
13:4	XFERSIZE	RW	-	Transfer Size (minus 1) 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. 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. The µDMA controller updates this field immediately prior to entering the arbitration process, so it contains the number of outstanding items that is necessary to complete the µDMA cycle.
3	NXTUSEBURST	RW	-	Next Useburst This field controls whether the Useburst SET[n] bit is automatically set for the last transfer of a peripheral scatter-gather operation. Normally, for the last transfer, if the number of remaining items to transfer is less than the arbitration size, the µDMA controller uses single transfers to complete the transaction. If this bit is set, then the controller uses a burst transfer to complete the last transfer.

Bit/Field	Name	Type	Reset	Description
2:0	XFERMODE	RW	-	μDMA Transfer Mode This field configures the operating mode of the μDMA cycle. Refer to "Transfer Modes" on page 523 for a detailed explanation of transfer modes. 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.

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 523.

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 524.

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 528.

Alternate Peripheral Scatter-Gather

This value must be used in the alternate channel control data structure when the μDMA controller operates in Peripheral Scatter-Gather mode.

8.6 μDMA Register Descriptions

The register addresses given are relative to the μDMA base address of 0x400F.F000.

Register 4: DMA Status (DMASTAT), offset 0x000

The **DMA Status (DMASTAT)** register returns the status of the μDMA controller. You cannot read this register when the μDMA controller is in the reset state.

DMA Status (DMASTAT)

Base 0x400F.F000
Offset 0x000
Type RO, reset 0x001F.0000

	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 μDMA Channels Minus 1 This field contains a value equal to the number of μDMA channels the μDMA controller is configured to use, minus one. The value of 0x1F corresponds to 32 μDMA channels.
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:4	STATE	RO	0x0	Control State Machine Status This field shows the current status of the control state machine. Status can be one of the following.

Value	Description
0x0	Idle
0x1	Reading channel controller data.
0x2	Reading source end pointer.
0x3	Reading destination end pointer.
0x4	Reading source data.
0x5	Writing destination data.
0x6	Waiting for μDMA request to clear.
0x7	Writing channel controller data.
0x8	Stalled
0x9	Done
0xA-0xF	Undefined

Bit/Field	Name	Type	Reset	Description
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 µDMA controller is disabled.
1	The µDMA controller is enabled.

Register 5: DMA Configuration (DMACFG), offset 0x004

The **DMACFG** register controls the configuration of the μDMA controller.

DMA Configuration (DMACFG)

Base 0x400F.F000
Offset 0x004
Type WO, reset -

	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															MASTEN
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 μDMA controller. 1 Enables μDMA controller.

Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008

The **DMACTLBASE** register must be configured so that the base pointer points to a location in system memory.

The amount of system memory that must be assigned to the μDMA controller depends on the number of μDMA channels used and whether the alternate channel control data structure is used. See “Channel Configuration” on page 521 for details about the Channel Control Table. The base address must be aligned on a 1024-byte boundary. This register cannot be read when the μDMA controller is in the reset state.

DMA Channel Control Base Pointer (DMACTLBASE)

Base 0x400F.F000
Offset 0x008
Type RW, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADDR						reserved									
Type	RW	RW	RW	RW	RW	RW	RO	RO	RO	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	ADDR	RW	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 μDMA 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 µDMA channel is waiting on a request. A peripheral can hold off the µDMA from performing a single request until the peripheral is ready for a burst request to enhance the µDMA performance. The use of this feature is dependent on the design of the peripheral and is not controllable by software in any way. This register cannot be read when the µDMA controller is in the reset state.

DMA Channel Wait-on-Request Status (DMAWAITSTAT)

Base 0x400F.F000
Offset 0x010
Type RO, reset 0x03C3.CF00

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WAITREQ[n]															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WAITREQ[n]															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	WAITREQ[n]	RO	0x03C3.CF00	Channel [n] Wait Status These bits provide the channel wait-on-request status. Bit 0 corresponds to channel 0.

Value	Description
0	The corresponding channel is not waiting on a request.
1	The corresponding channel is waiting on a request.

Register 9: DMA Channel Software Request (DMASWREQ), offset 0x014

Each bit of the **DMASWREQ** register represents the corresponding μDMA channel. Setting a bit generates a request for the specified μDMA channel.

DMA Channel Software Request (DMASWREQ)

Base 0x400F.F000
Offset 0x014
Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SWREQ[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
	SWREQ[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	SWREQ[n]	WO	-	Channel [n] Software Request These bits generate software requests. Bit 0 corresponds to channel 0. Value Description 0 No request generated. 1 Generate a software request for the corresponding channel. These bits are automatically cleared when the software request has been completed.

Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018

Each bit of the **DMAUSEBURSTSET** register represents the corresponding μDMA channel. Setting a bit disables the channel's single request input from generating requests, configuring the channel to only accept burst requests. Reading the register returns the status of USEBURST.

If the amount of data to transfer is a multiple of the arbitration (burst) size, the corresponding `SET[n]` bit is cleared after completing the final transfer. If there are fewer items remaining to transfer than the arbitration (burst) size, the μDMA controller automatically clears the corresponding `SET[n]` bit, allowing the remaining items to transfer using single requests. In order to resume transfers using burst requests, the corresponding bit must be set again. A bit should not be set if the corresponding peripheral does not support the burst request model.

Refer to “Request Types” on page 520 for more details about request types.

DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000
Offset 0x018
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	RW	0x0000.0000	Channel [n] Useburst Set Value Description 0 μDMA channel [n] responds to single or burst requests. 1 μDMA channel [n] responds only to burst requests. Bit 0 corresponds to channel 0. This bit is automatically cleared as described above. A bit can also be manually cleared by setting the corresponding `CLR[n]` bit in the **DMAUSEBURSTCLR** register.

Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C

Each bit of the **DMAUSEBURSTCLR** register represents the corresponding μDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAUSEBURSTSET** register.

DMA Channel Useburst Clear (DMAUSEBURSTCLR)

Base 0x400F.F000
Offset 0x01C
Type WO, reset -

	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 μDMA channel [n] responds to single and burst requests.

Register 12: DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020

Each bit of the **DMAREQMASKSET** register represents the corresponding μDMA channel. Setting a bit disables μDMA requests for the channel. Reading the register returns the request mask status. When a μDMA channel's request is masked, that means the peripheral can no longer request μDMA transfers. The channel can then be used for software-initiated transfers.

DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000
Offset 0x020
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	RW	0x0000.0000	Channel [n] Request Mask Set Value Description 0 The peripheral associated with channel [n] is enabled to request μDMA transfers. 1 The peripheral associated with channel [n] is not able to request μDMA transfers. Channel [n] may be used for software-initiated transfers. Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the **DMAREQMASKCLR** register.

Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024

Each bit of the **DMAREQMASKCLR** register represents the corresponding μDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAREQMASKSET** register.

DMA Channel Request Mask Clear (DMAREQMASKCLR)

Base 0x400F.F000
Offset 0x024
Type WO, reset -

	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] Request Mask Clear

Value	Description
0	No effect.
1	Setting a bit clears the corresponding `SET[n]` bit in the **DMAREQMASKSET** register meaning that the peripheral associated with channel [n] is enabled to request μDMA transfers.

Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028

Each bit of the **DMAENASET** register represents the corresponding μDMA channel. Setting a bit enables the corresponding μDMA channel. Reading the register returns the enable status of the channels. If a channel is enabled but the request mask is set (**DMAREQMASKSET**), then the channel can be used for software-initiated transfers.

DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000
Offset 0x028
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	RW	0x0000.0000	Channel [n] Enable Set Value Description 0 μDMA Channel [n] is disabled. 1 μDMA Channel [n] is enabled. Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAENACLR** register or when the end of a μDMA transfer occurs.

Register 15: DMA Channel Enable Clear (DMAENACLR), offset 0x02C

Each bit of the **DMAENACLR** register represents the corresponding μDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAENASET** register.

DMA Channel Enable Clear (DMAENACLR)

Base 0x400F.F000
Offset 0x02C
Type WO, reset -

	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 μDMA transfers.

Note: The controller disables a channel when it completes the μDMA cycle.

Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030

Each bit of the **DMAALTSET** register represents the corresponding μDMA channel. Setting a bit configures the μDMA channel to use the alternate control data structure. Reading the register returns the status of which control data structure is in use for the corresponding μDMA channel.

DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000
Offset 0x030
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	RW	0x0000.0000	Channel [n] Alternate Set

Value	Description
0	μDMA channel [n] is using the primary control structure.
1	μDMA channel [n] is using the alternate control structure.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAALTCLR** register.

Note: For Ping-Pong and Scatter-Gather cycle types, the μDMA controller automatically sets these bits to select the alternate channel control data structure.

Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034

Each bit of the **DMAALTCLR** register represents the corresponding μDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAALTSET** register.

DMA Channel Primary Alternate Clear (DMAALTCLR)

Base 0x400F.F000
Offset 0x034
Type WO, reset -

	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] Alternate Clear Value Description 0 No effect. 1 Setting a bit clears the corresponding `SET[n]` bit in the **DMAALTSET** register meaning that channel [n] is using the primary control structure. **Note:** For Ping-Pong and Scatter-Gather cycle types, the μDMA controller automatically sets these bits to select the alternate channel control data structure.

Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038

Each bit of the **DMAPRIOSET** register represents the corresponding μDMA channel. Setting a bit configures the μDMA channel to have a high priority level. Reading the register returns the status of the channel priority mask.

DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000
Offset 0x038
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	RW	0x0000.0000	Channel [n] Priority Set

Value	Description
0	μDMA channel [n] is using the default priority level.
1	μDMA channel [n] is using a high priority level.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAPRIOCLR** register.

Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C

Each bit of the **DMAPRIOCLR** register represents the corresponding μDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAPRIOSET** register.

DMA Channel Priority Clear (DMAPRIOCLR)

Base 0x400F.F000
Offset 0x03C
Type WO, reset -

	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] Priority Clear

Value	Description
0	No effect.
1	Setting a bit clears the corresponding `SET[n]` bit in the **DMAPRIOSET** register meaning that channel [n] is using the default priority level.

Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C

The **DMAERRCLR** register is used to read and clear the µDMA bus error status. The error status is set if the µDMA controller encountered a bus error while performing a transfer. If a bus error occurs on a channel, that channel is automatically disabled by the µDMA controller. The other channels are unaffected.

DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000
Offset 0x04C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ERRCLR	RW1C	0	µDMA Bus Error Status

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 Assignment (DMACHASGN), offset 0x500

Each bit of the **DMACHASGN** register represents the corresponding μDMA channel. Setting a bit selects the secondary channel assignment as specified in Table 8-1 on page 519.

Note: This register is provided to support legacy software. New software should use the **DMACHMAPn** registers. If a bit is clear in this register, the corresponding field in the **DMACHMAPn** registers is configured to 0x0. If a bit is set in this register, the corresponding field is configured to 0x1. If this register is read, a bit reads as 0 if the corresponding **DMACHMAPn** register field value is equal to 0, otherwise it reads as 1 if the corresponding **DMACHMAPn** register field value is not equal to 0.

DMA Channel Assignment (DMACHASGN)

Base 0x400F.F000
Offset 0x500
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CHASGN[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHASGN[n]															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	CHASGN[n]	RW	-	Channel [n] Assignment Select

Value	Description
0	Use the primary channel assignment.
1	Use the secondary channel assignment.

Register 22: DMA Channel Interrupt Status (DMACHIS), offset 0x504

Each bit of the **DMACHIS** register represents the corresponding μDMA channel. A bit is set when that μDMA channel causes a completion interrupt. The bits are cleared by a writing a 1.

Note: When transfers are performed from a FIFO of the UART or SSI using the μDMA, and any interrupt is generated from the UART or SSI, the module's status bit in the **DMACHIS** register must be checked at the end of the interrupt service routine. If the status bit is set, clear the interrupt by writing a 1 to it.

DMA Channel Interrupt Status (DMACHIS)

Base 0x400F.F000
Offset 0x504
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CHIS[n]															
Type	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHIS[n]															
Type	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	CHIS[n]	RW1C	0x0000.0000	Channel [n] Interrupt Status Value Description 0 The corresponding μDMA channel has not caused an interrupt. 1 The corresponding μDMA channel caused an interrupt. This bit is cleared by writing a 1 to it.

Register 23: DMA Channel Map Select 0 (DMACHMAP0), offset 0x510

Each 4-bit field of the **DMACHMAP0** register configures the μDMA channel assignment as specified in Table 8-1 on page 519.

Note: To support legacy software which uses the **DMA Channel Assignment (DMACHASGN)** register, a value of 0x0 is equivalent to a **DMACHASGN** bit being clear, and a value of 0x1 is equivalent to a **DMACHASGN** bit being set.

DMA Channel Map Select 0 (DMACHMAP0)

Base 0x400F.F000
Offset 0x510
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CH7SEL				CH6SEL				CH5SEL				CH4SEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CH3SEL				CH2SEL				CH1SEL				CH0SEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:28	CH7SEL	RW	0x00	μDMA Channel 7 Source Select See Table 8-1 on page 519 for channel assignments.
27:24	CH6SEL	RW	0x00	μDMA Channel 6 Source Select See Table 8-1 on page 519 for channel assignments.
23:20	CH5SEL	RW	0x00	μDMA Channel 5 Source Select See Table 8-1 on page 519 for channel assignments.
19:16	CH4SEL	RW	0x00	μDMA Channel 4 Source Select See Table 8-1 on page 519 for channel assignments.
15:12	CH3SEL	RW	0x00	μDMA Channel 3 Source Select See Table 8-1 on page 519 for channel assignments.
11:8	CH2SEL	RW	0x00	μDMA Channel 2 Source Select See Table 8-1 on page 519 for channel assignments.
7:4	CH1SEL	RW	0x00	μDMA Channel 1 Source Select See Table 8-1 on page 519 for channel assignments.
3:0	CH0SEL	RW	0x00	μDMA Channel 0 Source Select See Table 8-1 on page 519 for channel assignments.

Register 24: DMA Channel Map Select 1 (DMACHMAP1), offset 0x514

Each 4-bit field of the **DMACHMAP1** register configures the µDMA channel assignment as specified in Table 8-1 on page 519.

Note: To support legacy software which uses the **DMA Channel Assignment (DMACHASGN)** register, a value of 0x0 is equivalent to a **DMACHASGN** bit being clear, and a value of 0x1 is equivalent to a **DMACHASGN** bit being set.

DMA Channel Map Select 1 (DMACHMAP1)

Base 0x400F.F000
Offset 0x514
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CH15SEL				CH14SEL				CH13SEL				CH12SEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CH11SEL				CH10SEL				CH9SEL				CH8SEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:28	CH15SEL	RW	0x00	µDMA Channel 15 Source Select See Table 8-1 on page 519 for channel assignments.
27:24	CH14SEL	RW	0x00	µDMA Channel 14 Source Select See Table 8-1 on page 519 for channel assignments.
23:20	CH13SEL	RW	0x00	µDMA Channel 13 Source Select See Table 8-1 on page 519 for channel assignments.
19:16	CH12SEL	RW	0x00	µDMA Channel 12 Source Select See Table 8-1 on page 519 for channel assignments.
15:12	CH11SEL	RW	0x00	µDMA Channel 11 Source Select See Table 8-1 on page 519 for channel assignments.
11:8	CH10SEL	RW	0x00	µDMA Channel 10 Source Select See Table 8-1 on page 519 for channel assignments.
7:4	CH9SEL	RW	0x00	µDMA Channel 9 Source Select See Table 8-1 on page 519 for channel assignments.
3:0	CH8SEL	RW	0x00	µDMA Channel 8 Source Select See Table 8-1 on page 519 for channel assignments.

Register 25: DMA Channel Map Select 2 (DMACHMAP2), offset 0x518

Each 4-bit field of the **DMACHMAP2** register configures the μDMA channel assignment as specified in Table 8-1 on page 519.

Note: To support legacy software which uses the **DMA Channel Assignment (DMACHASGN)** register, a value of 0x0 is equivalent to a **DMACHASGN** bit being clear, and a value of 0x1 is equivalent to a **DMACHASGN** bit being set.

DMA Channel Map Select 2 (DMACHMAP2)

Base 0x400F.F000
Offset 0x518
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CH23SEL				CH22SEL				CH21SEL				CH20SEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CH19SEL				CH18SEL				CH17SEL				CH16SEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:28	CH23SEL	RW	0x00	μDMA Channel 23 Source Select See Table 8-1 on page 519 for channel assignments.
27:24	CH22SEL	RW	0x00	μDMA Channel 22 Source Select See Table 8-1 on page 519 for channel assignments.
23:20	CH21SEL	RW	0x00	μDMA Channel 21 Source Select See Table 8-1 on page 519 for channel assignments.
19:16	CH20SEL	RW	0x00	μDMA Channel 20 Source Select See Table 8-1 on page 519 for channel assignments.
15:12	CH19SEL	RW	0x00	μDMA Channel 19 Source Select See Table 8-1 on page 519 for channel assignments.
11:8	CH18SEL	RW	0x00	μDMA Channel 18 Source Select See Table 8-1 on page 519 for channel assignments.
7:4	CH17SEL	RW	0x00	μDMA Channel 17 Source Select See Table 8-1 on page 519 for channel assignments.
3:0	CH16SEL	RW	0x00	μDMA Channel 16 Source Select See Table 8-1 on page 519 for channel assignments.

Register 26: DMA Channel Map Select 3 (DMACHMAP3), offset 0x51C

Each 4-bit field of the **DMACHMAP3** register configures the µDMA channel assignment as specified in Table 8-1 on page 519.

Note: To support legacy software which uses the **DMA Channel Assignment (DMACHASGN)** register, a value of 0x0 is equivalent to a **DMACHASGN** bit being clear, and a value of 0x1 is equivalent to a **DMACHASGN** bit being set.

DMA Channel Map Select 3 (DMACHMAP3)

Base 0x400F.F000
Offset 0x51C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CH31SEL				CH30SEL				CH29SEL				CH28SEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CH27SEL				CH26SEL				CH25SEL				CH24SEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:28	CH31SEL	RW	0x00	µDMA Channel 31 Source Select See Table 8-1 on page 519 for channel assignments.
27:24	CH30SEL	RW	0x00	µDMA Channel 30 Source Select See Table 8-1 on page 519 for channel assignments.
23:20	CH29SEL	RW	0x00	µDMA Channel 29 Source Select See Table 8-1 on page 519 for channel assignments.
19:16	CH28SEL	RW	0x00	µDMA Channel 28 Source Select See Table 8-1 on page 519 for channel assignments.
15:12	CH27SEL	RW	0x00	µDMA Channel 27 Source Select See Table 8-1 on page 519 for channel assignments.
11:8	CH26SEL	RW	0x00	µDMA Channel 26 Source Select See Table 8-1 on page 519 for channel assignments.
7:4	CH25SEL	RW	0x00	µDMA Channel 25 Source Select See Table 8-1 on page 519 for channel assignments.
3:0	CH24SEL	RW	0x00	µDMA Channel 24 Source Select See Table 8-1 on page 519 for channel assignments.

Register 27: 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	μDMA Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 28: 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	μDMA Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 29: 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	µDMA Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 30: 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	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	PID3	RO	0x00	µDMA Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 31: 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	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	PID4	RO	0x04	μDMA Peripheral ID Register Can be used by software to identify the presence of this peripheral.

Register 32: 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	µDMA PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 33: 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	μDMA PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 34: 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	µDMA PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 35: 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	μDMA PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

9 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of seven physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, Port G). The GPIO module supports up to 49 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- Up to 49 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant in input configuration
- Ports A-G accessed through the Advanced Peripheral Bus (APB)
- Fast toggle capable of a change every clock cycle for ports on AHB, every two clock cycles for ports on APB
- 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 or a μDMA transfer
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can sink 18-mA for high-current applications
  - Slew rate control for 8-mA pad drive
  - Open drain enables
  - Digital input enables

9.1 Signal Description

GPIO signals have alternate hardware functions. The following table lists the GPIO pins and their analog and digital alternate functions. All GPIO signals are 5-V tolerant when configured as inputs except for PD4, PD5, PB0 and PB1, which are limited to 3.6 V. The digital alternate hardware functions are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the PMCx bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric encoding shown in the table below. Analog signals in the table below are also 5-V tolerant and are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. The AINx analog signals have internal circuitry to protect them from voltages over V_{DD} (up to the maximum specified in Table 21-1 on page 1122), but analog performance specifications are only guaranteed if the input signal swing at the I/O pad is kept inside the range 0 V < V_{IN} < V_{DD}. Note that each pin must be programmed individually; no type of grouping is implied by the columns in the table. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Important: The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ returns these GPIO to their original special consideration state.

Table 9-1. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	I^{21}C0	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO[a]	0	0	0	0	0x0	0
PF[0]	GPIO[a]	0	0	0	0	0x0	0

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see "Commit Control" on page 588.

Table 9-2. GPIO Pins and Alternate Functions (64LQFP)

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PA0	17	-	U0Rx	-	-	-	-	-	-	-	-	-	-
PA1	18	-	U0Tx	-	-	-	-	-	-	-	-	-	-
PA2	19	-	-	SSI0Clk	-	-	-	-	-	-	-	-	-
PA3	20	-	-	SSI0Fss	-	-	-	-	-	-	-	-	-
PA4	21	-	-	SSI0Rx	-	-	-	-	-	-	-	-	-
PA5	22	-	-	SSI0Tx	-	-	-	-	-	-	-	-	-
PA6	23	-	-	-	I2C1SCL	-	-	-	-	-	-	-	-
PA7	24	-	-	-	I2C1SDA	-	-	-	-	-	-	-	-
PB0	45	-	U1Rx	-	-	-	-	-	T2CCP0	-	-	-	-
PB1	46	-	U1Tx	-	-	-	-	-	T2CCP1	-	-	-	-
PB2	47	-	-	-	I2C0SCL	-	-	-	T3CCP0	-	-	-	-

Table 9-2. GPIO Pins and Alternate Functions (64LQFP) *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PB3	48	-	-	-	I2C0SDA	-	-	-	T3CCP1	-	-	-	-
PB4	58	AIN10	-	SSI2Clk	-	-	-	-	T1CCP0	CAN0Rx	-	-	-
PB5	57	AIN11	-	SSI2Fss	-	-	-	-	T1CCP1	CAN0Tx	-	-	-
PB6	1	-	-	SSI2Rx	I2C5SCL	-	-	-	T0CCP0	-	-	-	-
PB7	4	-	-	SSI2Tx	I2C5SDA	-	-	-	T0CCP1	-	-	-	-
PC0	52	-	TCK SWCLK	-	-	-	-	-	T4CCP0	-	-	-	-
PC1	51	-	TMS SWDIO	-	-	-	-	-	T4CCP1	-	-	-	-
PC2	50	-	TDI	-	-	-	-	-	T5CCP0	-	-	-	-
PC3	49	-	TDO SWO	-	-	-	-	-	T5CCP1	-	-	-	-
PC4	16	C1-	U4Rx	U1Rx	-	-	-	-	WT0CCP0	U1RTS	-	-	-
PC5	15	C1+	U4Tx	U1Tx	-	-	-	-	WT0CCP1	U1CTS	-	-	-
PC6	14	C0+	U3Rx	-	-	-	-	-	WT1CCP0	-	-	-	-
PC7	13	C0-	U3Tx	-	-	-	-	-	WT1CCP1	-	-	-	-
PD0	61	AIN7	SSI3Clk	SSI1Clk	I2C3SCL	-	-	-	WT2CCP0	-	-	-	-
PD1	62	AIN6	SSI3Fss	SSI1Fss	I2C3SDA	-	-	-	WT2CCP1	-	-	-	-
PD2	63	AIN5	SSI3Rx	SSI1Rx	-	-	-	-	WT3CCP0	-	-	-	-
PD3	64	AIN4	SSI3Tx	SSI1Tx	-	-	-	-	WT3CCP1	-	-	-	-
PD4	43	USB0DM	U6Rx	-	-	-	-	-	WT4CCP0	-	-	-	-
PD5	44	USB0DP	U6Tx	-	-	-	-	-	WT4CCP1	-	-	-	-
PD6	53	-	U2Rx	-	-	-	-	-	WT5CCP0	-	-	-	-
PD7	10	-	U2Tx	-	-	-	-	-	WT5CCP1	NMI	-	-	-
PE0	9	AIN3	U7Rx	-	-	-	-	-	-	-	-	-	-
PE1	8	AIN2	U7Tx	-	-	-	-	-	-	-	-	-	-
PE2	7	AIN1	-	-	-	-	-	-	-	-	-	-	-
PE3	6	AIN0	-	-	-	-	-	-	-	-	-	-	-
PE4	59	AIN9	U5Rx	-	I2C2SCL	-	-	-	-	CAN0Rx	-	-	-
PE5	60	AIN8	U5Tx	-	I2C2SDA	-	-	-	-	CAN0Tx	-	-	-
PF0	28	-	U1RTS	SSI1Rx	CAN0Rx	-	-	-	T0CCP0	NMI	C0o	-	-
PF1	29	-	U1CTS	SSI1Tx	-	-	-	-	T0CCP1	-	C1o	TRD1	-
PF2	30	-	-	SSI1Clk	-	-	-	-	T1CCP0	-	-	TRD0	-
PF3	31	-	-	SSI1Fss	CAN0Tx	-	-	-	T1CCP1	-	-	TRCLK	-
PF4	5	-	-	-	-	-	-	-	T2CCP0	-	-	-	-
PG0	37	-	-	-	I2C3SCL	-	-	-	T4CCP0	-	-	-	-
PG1	36	-	-	-	I2C3SDA	-	-	-	T4CCP1	-	-	-	-
PG2	35	-	-	-	I2C4SCL	-	-	-	T5CCP0	-	-	-	-

Table 9-2. GPIO Pins and Alternate Functions (64LQFP) *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PG3	34	-	-	-	I2C4SDA	-	-	-	T5CCP1	-	-	-	-
PG4	33	-	U2Rx	-	I2C1SCL	-	-	-	WT0CCP0	-	-	-	-
PG5	32	-	U2Tx	-	I2C1SDA	-	-	-	WT0CCP1	-	-	-	-

[a] The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin. Encodings 10-13 are not used on this device.

9.2 Functional Description

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 9-1 on page 584 and Figure 9-2 on page 585). The TM4C1232C3PM microcontroller contains seven ports and thus seven of these physical GPIO blocks. Note that not all pins are implemented on every block. Some GPIO pins can function as I/O signals for the on-chip peripheral modules. For information on which GPIO pins are used for alternate hardware functions, refer to Table 20-5 on page 1115.

Figure 9-1. Digital I/O Pads

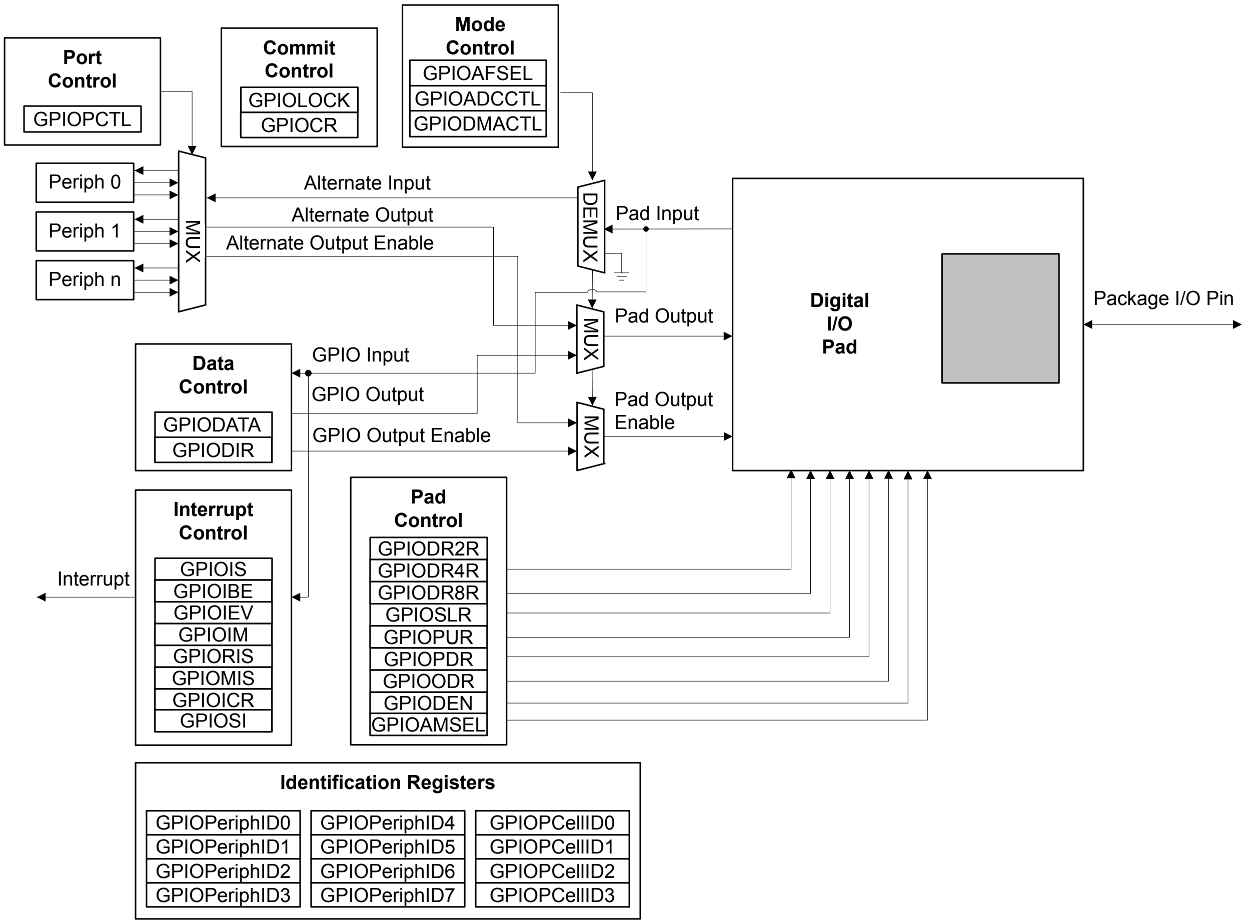
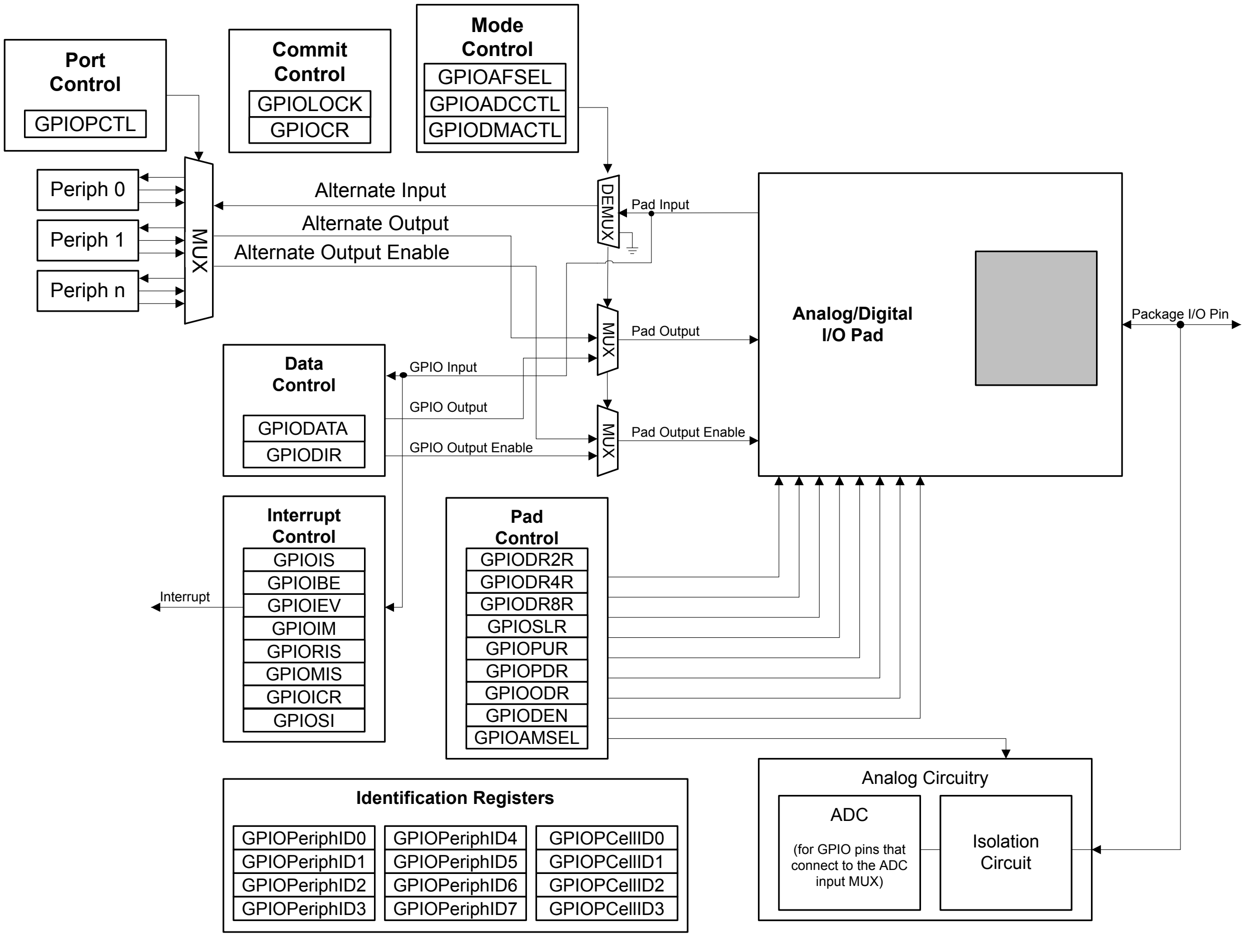


Figure 9-2. Analog/Digital I/O Pads



9.2.1 Data Control

The data control registers allow software to configure the operational modes of the GPIOs. The data direction register configures the GPIO as an input or an output while the data register either captures incoming data or drives it out to the pads.

Caution – It is possible to create a software sequence that prevents the debugger from connecting to the TM4C1232C3PM microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger. In the case that the software routine is not implemented and the device is locked out of the part, this issue can be solved by using the TM4C1232C3PM Flash Programmer "Unlock" feature. Please refer to LMFLASHPROGRAMMER on the TI web for more information.

9.2.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 595) is used to configure each individual pin as an input or output. When the data direction bit is cleared, the GPIO is configured as an input, and the corresponding data register bit captures and stores the value on the GPIO port. When the data direction bit is set, the GPIO is configured as an output, and the corresponding data register bit is driven out on the GPIO port.

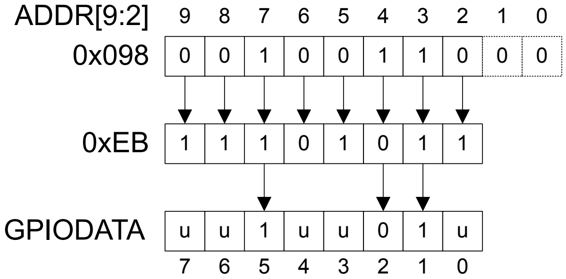
9.2.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 594) by using bits [9:2] of the address bus as a mask. In this manner, software drivers can modify individual GPIO pins in a single instruction without affecting the state of the other pins. This method is more efficient than the conventional method of performing a read-modify-write operation to set or clear an individual GPIO pin. To implement this feature, the **GPIODATA** register covers 256 locations in the memory map.

During a write, if the address bit associated with that data bit is set, the value of the **GPIODATA** register is altered. If the address bit is cleared, the data bit is left unchanged.

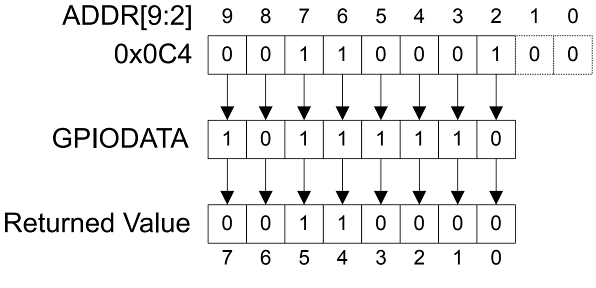
For example, writing a value of 0xEB to the address GPIODATA + 0x098 has the results shown in Figure 9-3, where u indicates that data is unchanged by the write. This example demonstrates how **GPIODATA** bits 5, 2, and 1 are written.

Figure 9-3. GPIODATA Write Example



During a read, if the address bit associated with the data bit is set, the value is read. If the address bit associated with the data bit is cleared, the data bit is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 9-4. This example shows how to read **GPIODATA** bits 5, 4, and 0.

Figure 9-4. GPIODATA Read Example



9.2.2 Interrupt Control

The interrupt capabilities of each GPIO port are controlled by a set of seven registers. These registers are used to select the source of the interrupt, its polarity, and the edge properties. When one or more GPIO inputs cause an interrupt, a single interrupt output is sent to the interrupt controller for the entire GPIO port. For edge-triggered interrupts, software must clear the interrupt to enable any further interrupts. For a level-sensitive interrupt, the external source must hold the level constant for the interrupt to be recognized by the controller.

Three registers define the edge or sense that causes interrupts:

- **GPIO Interrupt Sense (GPIOIS)** register (see page 596)
- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 597)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 598)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 599).

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 600 and page 601). As the name implies, the **GPIOMIS** register only shows interrupt conditions that are allowed to be passed to the interrupt controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the interrupt controller.

For a GPIO level-detect interrupt, the interrupt signal generating the interrupt must be held until serviced. Once the input signal deasserts from the interrupt generating logical sense, the corresponding RIS bit in the **GPIORIS** register clears. For a GPIO edge-detect interrupt, the RIS bit in the **GPIORIS** register is cleared by writing a ‘1’ to the corresponding bit in the **GPIO Interrupt Clear (GPIOICR)** register (see page 602). The corresponding **GPIOMIS** bit reflects the masked value of the RIS bit.

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

9.2.2.1 ADC Trigger Source

Any GPIO pin can be configured to be an external trigger for the ADC using the **GPIO ADC Control (GPIOADCCTL)** register. If any GPIO is configured as a non-masked interrupt pin (the appropriate bit of **GPIOIM** is set), and an interrupt for that port is generated, a trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated. See page 765.

Note that if the Port B **GPIOADCCTL** register is cleared, PB4 can still be used as an external trigger for the ADC. This is a legacy mode which allows code written for previous devices to operate on this microcontroller.

9.2.2.2 µDMA Trigger Source

Any GPIO pin can be configured to be an external trigger for the µDMA using the **GPIO DMA Control (GPIODMACTL)** register. If any GPIO is configured as a non-masked interrupt pin (the appropriate bit of **GPIOIM** is set), an interrupt for that port is generated and an external trigger signal is sent to the µDMA. If the µDMA is configured to start a transfer based on the GPIO signal, a transfer is initiated.

9.2.3 Mode Control

The GPIO pins can be controlled by either software or hardware. Software control is the default for most signals and corresponds to the GPIO mode, where the **GPIODATA** register is used to read or write the corresponding pins. When hardware control is enabled via the **GPIO Alternate Function**

Select (GPIOAFSEL) register (see page 603), the pin state is controlled by its alternate function (that is, the peripheral).

Further pin muxing options are provided through the **GPIO Port Control (GPIOPCTL)** register which selects one of several peripheral functions for each GPIO. For information on the configuration options, refer to Table 20-5 on page 1115.

Note: If any pin is to be used as an ADC input, the appropriate bit in the **GPIOAMSEL** register must be set to disable the analog isolation circuit.

9.2.4 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see “Signal Tables” on page 1099 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 603), **GPIO Pull Up Select (GPIOPUR)** register (see page 609), **GPIO Pull-Down Select (GPIOPDR)** register (see page 611), and **GPIO Digital Enable (GPIODEN)** register (see page 614) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 616) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 617) have been set.

9.2.5 Pad Control

The pad control registers allow software to configure the GPIO pads based on the application requirements. The pad control registers include the **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIOODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR**, and **GPIODEN** registers. These registers control drive strength, open-drain configuration, pull-up and pull-down resistors, slew-rate control and digital input enable for each GPIO. If 5 V is applied to a GPIO configured as an open-drain output, the output voltage will depend on the strength of your pull-up resistor. The GPIO pad is not electrically configured to output 5 V.

9.2.6 Identification

The identification registers configured at reset allow software to detect and identify the module as a GPIO block. The identification registers include the **GPIOPeriphID0**-**GPIOPeriphID7** registers as well as the **GPIOPCellID0**-**GPIOPCellID3** registers.

9.3 Initialization and Configuration

The GPIO modules may be accessed via two different memory apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous devices. The other aperture, the Advanced High-Performance Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus. These apertures are mutually exclusive. The aperture enabled for a given GPIO port is controlled by the appropriate bit in the **GPIOHBCTL** register (see page 245). Note that GPIO can only be accessed through the AHB aperture.

To configure the GPIO pins of a particular port, follow these steps:

1. Enable the clock to the port by setting the appropriate bits in the **RCGCGPIO** register (see page 319). In addition, the **SCGCGPIO** and **DCGCGPIO** registers can be programmed in the same manner to enable clocking in Sleep and Deep-Sleep modes.

2. Set the direction of the GPIO port pins by programming the **GPIODIR** register. A write of a 1 indicates output and a write of a 0 indicates input.

3. Configure the **GPIOAFSEL** register to program each bit as a GPIO or alternate pin. If an alternate pin is chosen for a bit, then the PMCx field must be programmed in the **GPIOPCTL** register for the specific peripheral required. There are also two registers, **GPIOADCCTL** and **GPIODMACTL**, which can be used to program a GPIO pin as a ADC or μDMA trigger, respectively.

4. Set the drive strength for each of the pins through the **GPIODR2R**, **GPIODR4R**, and **GPIODR8R** registers.

5. Program each pad in the port to have either pull-up, pull-down, or open drain functionality through the **GPIOPUR**, **GPIOPDR**, **GPIOODR** register. Slew rate may also be programmed, if needed, through the **GPIOSLR** register.

6. To enable GPIO pins as digital I/Os, set the appropriate DEN bit in the **GPIODEN** register. To enable GPIO pins to their analog function (if available), set the GPIOAMSEL bit in the **GPIOAMSEL** register.

7. Program the **GPIOIS**, **GPIOIBE**, **GPIOEV**, and **GPIOIM** registers to configure the type, event, and mask of the interrupts for each port.

   Note: To prevent false interrupts, the following steps should be taken when re-configuring GPIO edge and interrupt sense registers:

   a. Mask the corresponding port by clearing the IME field in the **GPIOIM** register.

   b. Configure the IS field in the **GPIOIS** register and the IBE field in the **GPIOIBE** register.

   c. Clear the **GPIORIS** register.

   d. Unmask the port by setting the IME field in the **GPIOIM** register.

8. Optionally, software can lock the configurations of the NMI and JTAG/SWD pins on the GPIO port pins, by setting the LOCK bits in the **GPIOLOCK** register.

When the internal POR signal is asserted and until otherwise configured, all GPIO pins are configured to be undriven (tristate): **GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0, except for the pins shown in Table 9-1 on page 582. Table 9-3 on page 589 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 9-4 on page 590 shows how a rising edge interrupt is configured for pin 2 of a GPIO port.

Table 9-3. 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 (I2CSDA)	1	X	1	1	X	X	?	?	?	?
Digital Input/Output (I2CSCL)	1	X	0	1	X	X	?	?	?	?
Digital Input (Timer CCP)	1	X	0	1	?	?	X	X	X	X

Table 9-3. 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 9-4. 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)

9.4 Register Map

Table 9-6 on page 592 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 devices. The other aperture, the Advanced High-Performance Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus.

Important: The GPIO registers in this chapter are duplicated in each GPIO block; however, depending on the block, all eight bits may not be connected to a GPIO pad. In those cases, writing to unconnected bits has no effect, and reading unconnected bits returns no meaningful data. See "Signal Description" on page 581 for the GPIOs included on this device.

The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A (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

Note that each GPIO module clock must be enabled before the registers can be programmed (see page 319). There must be a delay of 3 system clocks after the GPIO module clock is enabled before any GPIO module registers are accessed.

Important: The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ returns these GPIO to their original special consideration state.

Table 9-5. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	I^{21}C0	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO[a]	0	0	0	0	0x0	0
PF[0]	GPIO[a]	0	0	0	0	0x0	0

[a] This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 588.

The default register type for the **GPIOCR** register is RO for all GPIO pins with the exception of the NMI pin and the four JTAG/SWD pins (see “Signal Tables” on page 1099 for pin numbers). These six pins are the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for the corresponding GPIO Ports is RW.

The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the NMI and JTAG/SWD pins (see “Signal Tables” on page 1099 for pin numbers). To ensure that the JTAG and NMI pins are not accidentally programmed as GPIO pins, these pins default to non-committable. Because of this, the default reset value of **GPIOCR** changes for the corresponding ports.

Table 9-6. GPIO Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	RW	0x0000.0000	GPIO Data	594
0x400	GPIODIR	RW	0x0000.0000	GPIO Direction	595
0x404	GPIOIS	RW	0x0000.0000	GPIO Interrupt Sense	596
0x408	GPIOIBE	RW	0x0000.0000	GPIO Interrupt Both Edges	597
0x40C	GPIOIEV	RW	0x0000.0000	GPIO Interrupt Event	598
0x410	GPIOIM	RW	0x0000.0000	GPIO Interrupt Mask	599
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status	600
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	601
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	602
0x420	GPIOAFSEL	RW	-	GPIO Alternate Function Select	603
0x500	GPIODR2R	RW	0x0000.00FF	GPIO 2-mA Drive Select	605
0x504	GPIODR4R	RW	0x0000.0000	GPIO 4-mA Drive Select	606
0x508	GPIODR8R	RW	0x0000.0000	GPIO 8-mA Drive Select	607
0x50C	GPIOODR	RW	0x0000.0000	GPIO Open Drain Select	608
0x510	GPIOPUR	RW	-	GPIO Pull-Up Select	609
0x514	GPIOPDR	RW	0x0000.0000	GPIO Pull-Down Select	611
0x518	GPIOSLR	RW	0x0000.0000	GPIO Slew Rate Control Select	613
0x51C	GPIODEN	RW	-	GPIO Digital Enable	614
0x520	GPIOLOCK	RW	0x0000.0001	GPIO Lock	616
0x524	GPIOCR	-	-	GPIO Commit	617
0x528	GPIOAMSEL	RW	0x0000.0000	GPIO Analog Mode Select	619
0x52C	GPIOPCTL	RW	-	GPIO Port Control	620
0x530	GPIOADCCTL	RW	0x0000.0000	GPIO ADC Control	622
0x534	GPIODMACTL	RW	0x0000.0000	GPIO DMA Control	623
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	624
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	625
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	626
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	627
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	628
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	629
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	630
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	631

Table 9-6. GPIO Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	632
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	633
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	634
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	635

9.5 Register Descriptions

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

Register 1: GPIO Data (GPIODATA), offset 0x000

The **GPIODATA** register is the data register. In software control mode, values written in the **GPIODATA** register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the **GPIO Direction (GPIODIR)** register (see page 595).

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
Offset 0x000
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	RW	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 586 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
Offset 0x400
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DIR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	RW	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.

Note: To prevent false interrupts, the following steps should be taken when re-configuring GPIO edge and interrupt sense registers:

1. Mask the corresponding port by clearing the `IME` field in the **GPIOIM** register.
2. Configure the `IS` field in the **GPIOIS** register and the `IBE` field in the **GPIOIBE** register.
3. Clear the **GPIORIS** register.
4. Unmask the port by setting the `IME` field in the **GPIOIM** register.

GPIO Interrupt Sense (GPIOIS)

GPIO Port A (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
Offset 0x404
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								IS							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IS	RW	0x00	GPIO Interrupt Sense Value Description 0 The edge on the corresponding pin is detected (edge-sensitive). 1 The level on the corresponding pin is detected (level-sensitive).

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

The **GPIOIBE** register allows both edges to cause interrupts. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 596) 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 598). Clearing a bit configures the pin to be controlled by the **GPIOIEV** register. All bits are cleared by a reset.

Note: To prevent false interrupts, the following steps should be taken when re-configuring GPIO edge and interrupt sense registers:

1. Mask the corresponding port by clearing the `IME` field in the **GPIOIM** register.
2. Configure the `IS` field in the **GPIOIS** register and the `IBE` field in the **GPIOIBE** register.
3. Clear the **GPIORIS** register.
4. Unmask the port by setting the `IME` field in the **GPIOIM** register.

GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A (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
Offset 0x408
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								IBE							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	RW	0x00	GPIO Interrupt Both Edges

Value	Description
0	Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 598).
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 596). 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
Offset 0x40C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								IEV							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IEV	RW	0x00	GPIO Interrupt Event

Value	Description
0	A falling edge or a Low level on the corresponding pin triggers an interrupt.
1	A rising edge or a High level on the corresponding pin triggers an interrupt.

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

The **GPIOIM** register is the interrupt mask register. Setting a bit in the **GPIOIM** register allows interrupts that are generated by the corresponding pin to be sent to the interrupt controller on the combined interrupt signal. Clearing a bit prevents an interrupt on the corresponding pin from being sent to the interrupt controller. All bits are cleared by a reset.

GPIO Interrupt Mask (GPIOIM)

GPIO Port A (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
Offset 0x410
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								IME							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IME	RW	0x00	GPIO Interrupt Mask Enable

Value	Description
0	The interrupt from the corresponding pin is masked.
1	The interrupt from the corresponding pin is sent to the interrupt controller.

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

The **GPIORIS** register is the raw interrupt status register. A bit in this register is set when an interrupt condition occurs on the corresponding GPIO pin. If the corresponding bit in the **GPIO Interrupt Mask (GPIOIM)** register (see page 599) is set, the interrupt is sent to the interrupt controller. Bits read as zero indicate that corresponding input pins have not initiated an interrupt. For a GPIO level-detect interrupt, the interrupt signal generating the interrupt must be held until serviced. Once the input signal deasserts from the interrupt generating logical sense, the corresponding `RIS` bit in the **GPIORIS** register clears. For a GPIO edge-detect interrupt, the `RIS` bit in the **GPIORIS** register is cleared by writing a '1' to the corresponding bit in the **GPIO Interrupt Clear (GPIOICR)** register. The corresponding **GPIOMIS** bit reflects the masked value of the `RIS` bit.

GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A (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
Offset 0x414
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
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	RIS	RO	0x00	GPIO Interrupt Raw Status Value Description 0 An interrupt condition has not occurred on the corresponding pin. 1 An interrupt condition has occurred on the corresponding pin. For edge-detect interrupts, this bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register. For a GPIO level-detect interrupt, the bit is cleared when the level is deasserted.

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

The **GPIOMIS** register is the masked interrupt status register. If a bit is set in this register, the corresponding interrupt has triggered an interrupt to the interrupt controller. If a bit is clear, either no interrupt has been generated, or the interrupt is masked.

Note that if the Port B **GPIOADCCTL** register is cleared, PB4 can still be used as an external trigger for the ADC. This is a legacy mode which allows code written for previous devices to operate on this microcontroller.

GPIOMIS is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A (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
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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	MIS	RO	0x00	GPIO Masked Interrupt Status Value Description 0 An interrupt condition on the corresponding pin is masked or has not occurred. 1 An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller. For edge-detect interrupts, this bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register. For a GPIO level-detect interrupt, the bit is cleared when the level is deasserted.

Register 9: GPIO Interrupt Clear (GPIOICR), offset 0x41C

The **GPIOICR** register is the interrupt clear register. For edge-detect interrupts, writing a 1 to the `IC` bit in the **GPIOICR** register clears the corresponding bit in the **GPIORIS** and **GPIOMIS** registers. If the interrupt is a level-detect, the `IC` bit in this register has no effect. In addition, writing a 0 to any of the bits in the **GPIOICR** register has no effect.

GPIO Interrupt Clear (GPIOICR)

GPIO Port A (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
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	0	Software should not rely on the value of a reserved 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 0 The corresponding interrupt is unaffected. 1 The corresponding interrupt is cleared.

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

The **GPIOAFSEL** register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral. Several possible peripheral functions are multiplexed on each GPIO. The **GPIO Port Control (GPIOPCTL)** register is used to select one of the possible functions. Table 20-5 on page 1115 details which functions are muxed on each GPIO pin. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

Important: The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ returns these GPIO to their original special consideration state.

Table 9-7. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	I^{21}C0	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO[a]	0	0	0	0	0x0	0
PF[0]	GPIO[a]	0	0	0	0	0x0	0

[a] This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 588.

Caution – It is possible to create a software sequence that prevents the debugger from connecting to the TM4C1232C3PM microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger. In the case that the software routine is not implemented and the device is locked out of the part, this issue can be solved by using the TM4C1232C3PM Flash Programmer "Unlock" feature. Please refer to LMFLASHPROGRAMMER on the TI web for more information.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see “Signal Tables” on page 1099 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 603), **GPIO Pull Up Select (GPIOPUR)** register (see page 609), **GPIO Pull-Down Select (GPIOPDR)** register (see page 611), and **GPIO Digital Enable (GPIODEN)** register (see page 614) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 616) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 617) have been set.

When using the I^{2}C module, in addition to setting the **GPIOAFSEL** register bits for the I^{2}C clock and data pins, the data pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register (see examples in “Initialization and Configuration” on page 588).

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
Offset 0x420
Type RW, 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	RW	RW	RW	RW	RW	RW	RW	RW
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	RW	-	GPIO Alternate Function Select Value Description 0 The associated pin functions as a GPIO and is controlled by the GPIO registers. 1 The associated pin functions as a peripheral signal and is controlled by the alternate hardware function. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 9-1 on page 582.

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
Offset 0x500
Type RW, reset 0x0000.00FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DRV2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV2	RW	0xFF	Output Pad 2-mA Drive Enable

Value	Description
0	The drive for the corresponding GPIO pin is controlled by the **GPIODR4R** or **GPIODR8R** register.
1	The corresponding GPIO pin has 2-mA drive.

Setting a bit in either the **GPIODR4** register or the **GPIODR8** register clears the corresponding 2-mA enable bit. The change is effective on the 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
Offset 0x504
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DRV4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV4	RW	0x00	Output Pad 4-mA Drive Enable Value Description 0 The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR8R** register. 1 The corresponding GPIO pin has 4-mA drive. Setting a bit in either the **GPIODR2** register or the **GPIODR8** register clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

The **GPIODR8R** register is the 8-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the DRV8 bit for a GPIO signal, the corresponding DRV2 bit in the **GPIODR2R** register and DRV4 bit in the **GPIODR4R** register are automatically cleared by hardware. The 8-mA setting is also used for high-current operation.

Note: There is no configuration difference between 8-mA and high-current operation. The additional current capacity results from a shift in the V_{OH}/V_{OL} levels. See “Recommended Operating Conditions” on page 1124 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
Offset 0x508
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DRV8							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV8	RW	0x00	Output Pad 8-mA Drive Enable

Value	Description
0	The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR4R** register.
1	The corresponding GPIO pin has 8-mA drive.

Setting a bit in either the **GPIODR2** register or the **GPIODR4** register clears the corresponding 8-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open-drain configuration of the corresponding GPIO pad. When open-drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Enable (GPIODEN)** register (see page 614). Corresponding bits in the drive strength and slew rate control registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired fall times. The GPIO acts as an input if the corresponding bit in the **GPIODIR** register is cleared. If open drain is selected while the GPIO is configured as an input, the GPIO will remain an input and the open-drain selection has no effect until the GPIO is changed to an output.

When using the I^{2}C module, in addition to configuring the data pin to open drain, the **GPIO Alternate Function Select (GPIOAFSEL)** register bits for the I^{2}C clock and data pins should be set (see examples in "Initialization and Configuration" on page 588).

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
Offset 0x50C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								ODE							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ODE	RW	0x00	Output Pad Open Drain Enable

Value	Description
0	The corresponding pin is not configured as open drain.
1	The corresponding pin is configured as open drain.

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

The **GPIOPUR** register is the pull-up control register. When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 611). Write access to this register is protected with the **GPIOCR** register. Bits in **GPIOCR** that are cleared prevent writes to the equivalent bit in this register.

Important: The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ returns these GPIO to their original special consideration state.

Table 9-8. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	I^{21}C0	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO[a]	0	0	0	0	0x0	0
PF[0]	GPIO[a]	0	0	0	0	0x0	0

[a] This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the `NMI` signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 588.

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the `NMI` pin (see “Signal Tables” on page 1099 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 603), **GPIO Pull Up Select (GPIOPUR)** register (see page 609), **GPIO Pull-Down Select (GPIOPDR)** register (see page 611), and **GPIO Digital Enable (GPIODEN)** register (see page 614) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 616) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 617) 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
Offset 0x510
Type RW, 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								PUE							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PUE	RW	-	Pad Weak Pull-Up Enable

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

Setting a bit in the **GPIOPDR** register clears the corresponding bit in the **GPIOPUR** register. The change is effective on the 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 9-1 on page 582.

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 609).

Important: The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ returns these GPIO to their original special consideration state.

Table 9-9. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	I^{21}C0	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO[a]	0	0	0	0	0x0	0
PF[0]	GPIO[a]	0	0	0	0	0x0	0

[a] This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 588.

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see “Signal Tables” on page 1099 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 603), **GPIO Pull Up Select (GPIOPUR)** register (see page 609), **GPIO Pull-Down Select (GPIOPDR)** register (see page 611), and **GPIO Digital Enable (GPIODEN)** register (see page 614) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 616) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 617) 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
Offset 0x514
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PDE							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PDE	RW	0x00	Pad Weak Pull-Down Enable

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

Setting a bit in the **GPIOPUR** register clears the corresponding bit in the **GPIOPDR** register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

Register 17: GPIO Slew Rate Control Select (GPIOSLR), offset 0x518

The **GPIOSLR** register is the slew rate control register. Slew rate control is only available when using the 8-mA drive strength option. The selection of drive strength is done through the **GPIO 8-mA Drive Select (GPIODR8R)** register.

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
Offset 0x518
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								SRL							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	SRL	RW	0x00	Slew Rate Limit Enable (8-mA drive only)

Value	Description
0	Slew rate control is disabled for the corresponding pin.
1	Slew rate control is enabled for the corresponding pin.

Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C

Note: Pins configured as digital inputs are Schmitt-triggered.

The **GPIODEN** register is the digital enable register. By default, all GPIO signals except those listed below are configured out of reset to be undriven (tristate). Their digital function is disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use the pin as a digital input or output (either GPIO or alternate function), the corresponding `GPIODEN` bit must be set.

Important: The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ returns these GPIO to their original special consideration state.

Table 9-10. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	I^{21}C0	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO[a]	0	0	0	0	0x0	0
PF[0]	GPIO[a]	0	0	0	0	0x0	0

[a] This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the `NMI` signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 588.

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the `NMI` pin (see “Signal Tables” on page 1099 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 603), **GPIO Pull Up Select (GPIOPUR)** register (see page 609), **GPIO Pull-Down Select (GPIOPDR)** register (see page 611), and **GPIO Digital Enable (GPIODEN)** register (see page 614) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 616) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 617) 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
Offset 0x51C
Type RW, 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	RW	RW	RW	RW	RW	RW	RW	RW
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	RW	-	Digital Enable

Value	Description
0	The digital functions for the corresponding pin are disabled.
1	The digital functions for the corresponding pin are enabled.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 9-1 on page 582.

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

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 617). 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
Offset 0x520
Type RW, reset 0x0000.0001

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

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

Bit/Field	Name	Type	Reset	Description
31:0	LOCK	RW	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
0x1	The **GPIOCR** register is locked and may not be modified.
0x0	The **GPIOCR** register is unlocked and may be modified.

Register 20: GPIO Commit (GPIOCR), offset 0x524

The **GPIOCR** register is the commit register. The value of the **GPIOCR** register determines which bits of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, and **GPIODEN** registers are committed when a write to these registers is performed. If a bit in the **GPIOCR** register is cleared, the data being written to the corresponding bit in the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** registers cannot be committed and retains its previous value. If a bit in the **GPIOCR** register is set, the data being written to the corresponding bit of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** registers is committed to the register and reflects the new value.

The contents of the **GPIOCR** register can only be modified if the status in the **GPIOLOCK** register is unlocked. Writes to the **GPIOCR** register are ignored if the status in the **GPIOLOCK** register is locked.

Important: This register is designed to prevent accidental programming of the registers that control connectivity to the NMI and JTAG/SWD debug hardware. By initializing the bits of the **GPIOCR** register to 0 for the NMI and JTAG/SWD pins (see "Signal Tables" on page 1099 for pin numbers), the NMI and JTAG/SWD debug port can only be converted to GPIOs through a deliberate set of writes to the **GPIOLOCK**, **GPIOCR**, and the corresponding registers.

Because this protection is currently only implemented on the NMI and JTAG/SWD pins (see "Signal Tables" on page 1099 for pin numbers), all of the other bits in the **GPIOCR** registers cannot be written with 0x0. These bits are hardwired to 0x1, ensuring that it is always possible to commit new values to the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** register bits of these other pins.

GPIO Commit (GPIOCR)

GPIO Port A (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
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.

Bit/Field	Name	Type	Reset	Description
7:0	CR	-	-	GPIO Commit Value Description 0 The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** bits cannot be written. 1 The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** bits can 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 (see “Signal Tables” on page 1099 for pin numbers). These six pins are the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for the corresponding GPIO Ports is RW. The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the `NMI` and JTAG/SWD pins (see “Signal Tables” on page 1099 for pin numbers). To ensure that the JTAG and `NMI` pins are not accidentally programmed as GPIO pins, these pins default to non-committable. Because of this, the default reset value of **GPIOCR** changes for the corresponding ports.

Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528

Important: This register is only valid for ports and pins that can be used as ADC AINx inputs.

If any pin is to be used as an ADC input, the appropriate bit in **GPIOAMSEL** must be set to disable the analog isolation circuit.

The **GPIOAMSEL** register controls isolation circuits to the analog side of a unified I/O pad. Because the GPIOs may be driven by a 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 20-5 on page 1115.

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
Offset 0x528
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								GPIOAMSEL							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	GPIOAMSEL	RW	0x00	GPIO Analog Mode Select Value Description 0 The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers. 1 The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions. **Note:** This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. The reset state of this register is 0 for all signals.

Register 22: GPIO Port Control (GPIOPCTL), offset 0x52C

The **GPIOPCTL** register is used in conjunction with the **GPIOAFSEL** register and selects the specific peripheral signal for each GPIO pin when using the alternate function mode. Most bits in the **GPIOAFSEL** register are cleared on reset, therefore most GPIO pins are configured as GPIOs by default. When a bit is set in the **GPIOAFSEL** register, the corresponding GPIO signal is controlled by an associated peripheral. The **GPIOPCTL** register selects one out of a set of peripheral functions for each GPIO, providing additional flexibility in signal definition. For information on the defined encodings for the bit fields in this register, refer to Table 20-5 on page 1115. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

Note: If a particular input signal to a peripheral is assigned to two different GPIO port pins, the signal is assigned to the port with the lowest letter and the assignment to the higher letter port is ignored. If a particular output signal from a peripheral is assigned to two different GPIO port pins, the signal will output to both pins. Assigning an output signal from a peripheral to two different GPIO pins is not recommended.

Important: The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ returns these GPIO to their original special consideration state.

Table 9-11. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	I^{21}C0	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO[a]	0	0	0	0	0x0	0
PF[0]	GPIO[a]	0	0	0	0	0x0	0

[a] This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 588.

GPIO Port Control (GPIOPCTL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
Offset 0x52C
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PMC7				PMC6				PMC5				PMC4			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PMC3				PMC2				PMC1				PMC0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:28	PMC7	RW	-	Port Mux Control 7 This field controls the configuration for GPIO pin 7.
27:24	PMC6	RW	-	Port Mux Control 6 This field controls the configuration for GPIO pin 6.
23:20	PMC5	RW	-	Port Mux Control 5 This field controls the configuration for GPIO pin 5.
19:16	PMC4	RW	-	Port Mux Control 4 This field controls the configuration for GPIO pin 4.
15:12	PMC3	RW	-	Port Mux Control 3 This field controls the configuration for GPIO pin 3.
11:8	PMC2	RW	-	Port Mux Control 2 This field controls the configuration for GPIO pin 2.
7:4	PMC1	RW	-	Port Mux Control 1 This field controls the configuration for GPIO pin 1.
3:0	PMC0	RW	-	Port Mux Control 0 This field controls the configuration for GPIO pin 0.

Register 23: GPIO ADC Control (GPIOADCCTL), offset 0x530

This register is used to configure a GPIO pin as a source for the ADC trigger.

Note that if the Port B **GPIOADCCTL** register is cleared, PB4 can still be used as an external trigger for the ADC. This is a legacy mode which allows code written for previous devices to operate on this microcontroller.

GPIO ADC Control (GPIOADCCTL)

GPIO Port A (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
Offset 0x530
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								ADCEN							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ADCEN	RW	0x00	ADC Trigger Enable

Value	Description
0	The corresponding pin is not used to trigger the ADC.
1	The corresponding pin is used to trigger the ADC.

Register 24: GPIO DMA Control (GPIODMACTL), offset 0x534

This register is used to configure a GPIO pin as a source for the µDMA trigger.

GPIO DMA Control (GPIODMACTL)

GPIO Port A (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
Offset 0x534
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DMAEN							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DMAEN	RW	0x00	µDMA Trigger Enable

Value	Description
0	The corresponding pin is not used to trigger the µDMA.
1	The corresponding pin is used to trigger the µDMA.

Register 25: GPIO 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
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	GPIO Peripheral ID Register [7:0]

Register 26: 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
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 27: 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
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	GPIO Peripheral ID Register [23:16]

Register 28: 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
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	GPIO Peripheral ID Register [31:24]

Register 29: 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
Offset 0xFE0
Type RO, reset 0x0000.0061

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	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	0x61	GPIO Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 30: 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
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	GPIO Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 31: 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
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	GPIO Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 32: 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
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 33: GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 0 (GPIOPCellID0)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
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	GPIO PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 34: GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 1 (GPIOPCellID1)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
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 35: GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 2 (GPIOPCellID2)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
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	GPIO PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 36: GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 3 (GPIOPCellID3)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
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.

10 General-Purpose Timers

Programmable timers can be used to count or time external events that drive the Timer input pins. The TM4C1232C3PM General-Purpose Timer Module (GPTM) contains six 16/32-bit GPTM blocks and six 32/64-bit Wide GPTM blocks. Each 16/32-bit GPTM block provides two 16-bit timers/counters (referred to as Timer A and Timer B) that can be configured to operate independently as timers or event counters, or concatenated to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Each 32/64-bit Wide GPTM block provides 32-bit timers for Timer A and Timer B that can be concatenated to operate as a 64-bit timer. Timers can also be used to trigger µDMA transfers.

In addition, timers can be used to trigger analog-to-digital conversions (ADC) when a time-out occurs in periodic and one-shot modes. The ADC trigger signals from all of the general-purpose timers are ORed together before reaching the ADC module, so only one timer should be used to trigger ADC events.

The GPT Module is one timing resource available on the Tiva™ C Series microcontrollers. Other timer resources include the System Timer (SysTick) (see 112).

The General-Purpose Timer Module (GPTM) contains six 16/32-bit GPTM blocks and six 32/64-bit Wide GPTM blocks with the following functional options:

- 16/32-bit operating modes:
  - 16- or 32-bit programmable one-shot timer
  - 16- or 32-bit programmable periodic timer
  - 16-bit general-purpose timer with an 8-bit prescaler
  - 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
  - 16-bit input-edge count- or time-capture modes with an 8-bit prescaler
  - 16-bit PWM mode with an 8-bit prescaler and software-programmable output inversion of the PWM signal
- 32/64-bit operating modes:
  - 32- or 64-bit programmable one-shot timer
  - 32- or 64-bit programmable periodic timer
  - 32-bit general-purpose timer with a 16-bit prescaler
  - 64-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
  - 32-bit input-edge count- or time-capture modes with a16-bit prescaler
  - 32-bit PWM mode with a 16-bit prescaler and software-programmable output inversion of the PWM signal
- Count up or down
- Twelve 16/32-bit Capture Compare PWM pins (CCP)
- Twelve 32/64-bit Capture Compare PWM pins (CCP)

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

10.1 Block Diagram

In the block diagram, the specific Capture Compare PWM (CCP) pins available depend on the TM4C1232C3PM device. See Table 10-1 on page 638 for the available CCP pins and their timer assignments.

Figure 10-1. GPTM Module Block Diagram

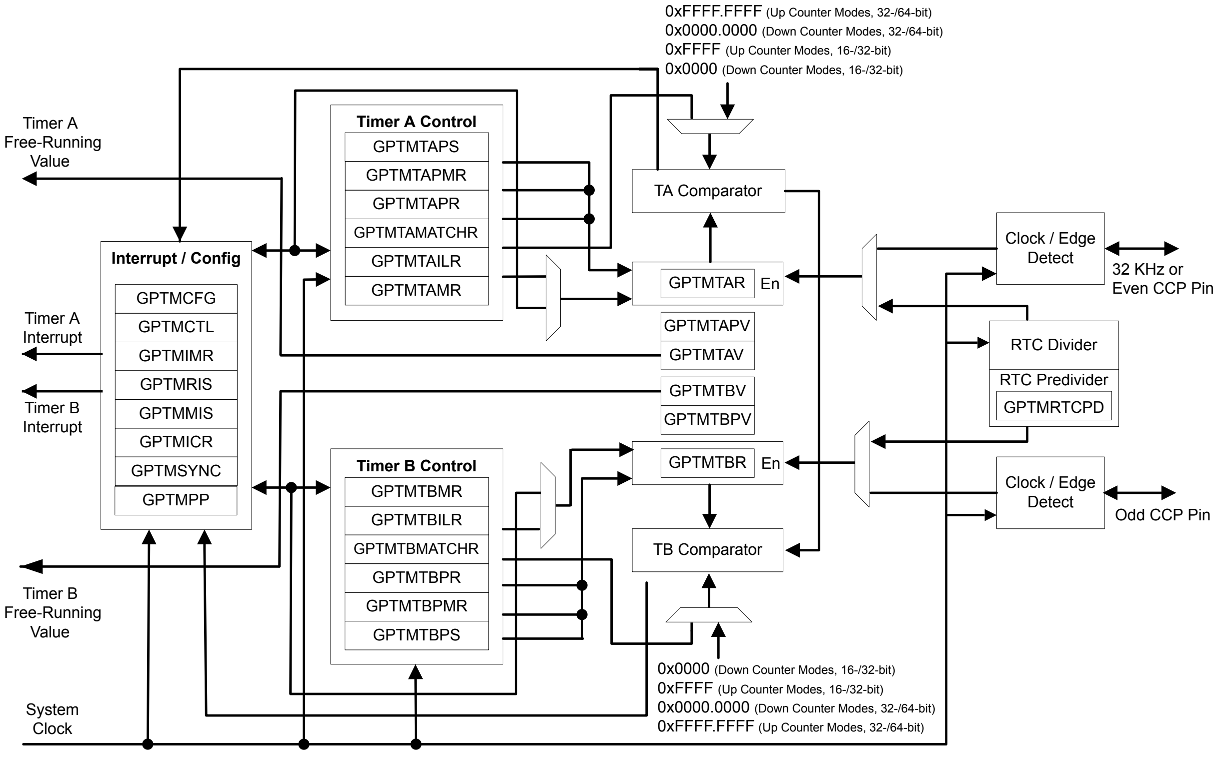


Table 10-1. Available CCP Pins

Timer	Up/Down Counter	Even CCP Pin	Odd CCP Pin
16/32-Bit Timer 0	Timer A	T0CCP0	-
	Timer B	-	T0CCP1
16/32-Bit Timer 1	Timer A	T1CCP0	-
	Timer B	-	T1CCP1
16/32-Bit Timer 2	Timer A	T2CCP0	-
	Timer B	-	T2CCP1
16/32-Bit Timer 3	Timer A	T3CCP0	-
	Timer B	-	T3CCP1
16/32-Bit Timer 4	Timer A	T4CCP0	-
	Timer B	-	T4CCP1
16/32-Bit Timer 5	Timer A	T5CCP0	-
	Timer B	-	T5CCP1
32/64-Bit Wide Timer 0	Timer A	WT0CCP0	-
	Timer B	-	WT0CCP1
32/64-Bit Wide Timer 1	Timer A	WT1CCP0	-
	Timer B	-	WT1CCP1
32/64-Bit Wide Timer 2	Timer A	WT2CCP0	-
	Timer B	-	WT2CCP1
32/64-Bit Wide Timer 3	Timer A	WT3CCP0	-
	Timer B	-	WT3CCP1
32/64-Bit Wide Timer 4	Timer A	WT4CCP0	-
	Timer B	-	WT4CCP1
32/64-Bit Wide Timer 5	Timer A	WT5CCP0	-
	Timer B	-	WT5CCP1

10.2 Signal Description

The following table lists the external signals of the GP Timer module and describes the function of each. The GP Timer signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these GP Timer signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 603) should be set to choose the GP Timer function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 620) 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 581.

Table 10-2. General-Purpose Timers Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
T0CCP0	1 28	PB6 (7) PF0 (7)	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 0.
T0CCP1	4 29	PB7 (7) PF1 (7)	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 1.

Table 10-2. General-Purpose Timers Signals (64LQFP) *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
T1CCP0	30 58	PF2 (7) PB4 (7)	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 0.
T1CCP1	31 57	PF3 (7) PB5 (7)	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 1.
T2CCP0	5 45	PF4 (7) PB0 (7)	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 0.
T2CCP1	46	PB1 (7)	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 1.
T3CCP0	47	PB2 (7)	I/O	TTL	16/32-Bit Timer 3 Capture/Compare/PWM 0.
T3CCP1	48	PB3 (7)	I/O	TTL	16/32-Bit Timer 3 Capture/Compare/PWM 1.
T4CCP0	37 52	PG0 (7) PC0 (7)	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 0.
T4CCP1	36 51	PG1 (7) PC1 (7)	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 1.
T5CCP0	35 50	PG2 (7) PC2 (7)	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 0.
T5CCP1	34 49	PG3 (7) PC3 (7)	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 1.
WT0CCP0	16 33	PC4 (7) PG4 (7)	I/O	TTL	32/64-Bit Wide Timer 0 Capture/Compare/PWM 0.
WT0CCP1	15 32	PC5 (7) PG5 (7)	I/O	TTL	32/64-Bit Wide Timer 0 Capture/Compare/PWM 1.
WT1CCP0	14	PC6 (7)	I/O	TTL	32/64-Bit Wide Timer 1 Capture/Compare/PWM 0.
WT1CCP1	13	PC7 (7)	I/O	TTL	32/64-Bit Wide Timer 1 Capture/Compare/PWM 1.
WT2CCP0	61	PD0 (7)	I/O	TTL	32/64-Bit Wide Timer 2 Capture/Compare/PWM 0.
WT2CCP1	62	PD1 (7)	I/O	TTL	32/64-Bit Wide Timer 2 Capture/Compare/PWM 1.
WT3CCP0	63	PD2 (7)	I/O	TTL	32/64-Bit Wide Timer 3 Capture/Compare/PWM 0.
WT3CCP1	64	PD3 (7)	I/O	TTL	32/64-Bit Wide Timer 3 Capture/Compare/PWM 1.
WT4CCP0	43	PD4 (7)	I/O	TTL	32/64-Bit Wide Timer 4 Capture/Compare/PWM 0.
WT4CCP1	44	PD5 (7)	I/O	TTL	32/64-Bit Wide Timer 4 Capture/Compare/PWM 1.
WT5CCP0	53	PD6 (7)	I/O	TTL	32/64-Bit Wide Timer 5 Capture/Compare/PWM 0.
WT5CCP1	10	PD7 (7)	I/O	TTL	32/64-Bit Wide Timer 5 Capture/Compare/PWM 1.

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

10.3 Functional Description

The main components of each GPTM block are two free-running up/down counters (referred to as Timer A and Timer B), two prescaler registers, two match registers, two prescaler match registers, two shadow registers, and two load/initialization registers and their associated control functions. The exact functionality of each GPTM is controlled by software and configured through the register interface. Timer A and Timer B can be used individually, in which case they have a 16-bit counting range for the 16/32-bit GPTM blocks and a 32-bit counting range for 32/64-bit Wide GPTM blocks. In addition, Timer A and Timer B can be concatenated to provide a 32-bit counting range for the 16/32-bit GPTM blocks and a 64-bit counting range for the 32/64-bit Wide GPTM blocks. Note that the prescaler can only be used when the timers are used individually.

The available modes for each GPTM block are shown in Table 10-3 on page 640. Note that when counting down in one-shot or periodic modes, the prescaler acts as a true prescaler and contains the least-significant bits of the count. When counting up in one-shot or periodic modes, the prescaler acts as a timer extension and holds the most-significant bits of the count. In input edge count, input edge time and PWM mode, the prescaler always acts as a timer extension, regardless of the count direction.

Table 10-3. General-Purpose Timer Capabilities

Mode	Timer Use	Count Direction	Counter Size		Prescaler Size[a]		Prescaler Behavior (Count Direction)
			16/32-bit GPTM	32/64-bit Wide GPTM	16/32-bit GPTM	32/64-bit Wide GPTM	
One-shot	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Up), Prescaler (Down)
	Concatenated	Up or Down	32-bit	64-bit	-	-	N/A
Periodic	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Up), Prescaler (Down)
	Concatenated	Up or Down	32-bit	64-bit	-	-	N/A
RTC	Concatenated	Up	32-bit	64-bit	-	-	N/A
Edge Count	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Both)
Edge Time	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Both)
PWM	Individual	Down	16-bit	32-bit	8-bit	16-bit	Timer Extension

a. The prescaler is only available when the timers are used individually

Software configures the GPTM using the **GPTM Configuration (GPTMCFG)** register (see page 659), the **GPTM Timer A Mode (GPTMTAMR)** register (see page 661), and the **GPTM Timer B Mode (GPTMTBMR)** register (see page 665). When in one of the concatenated modes, Timer A and Timer B can only operate in one mode. However, when configured in an individual mode, Timer A and Timer B can be independently configured in any combination of the individual modes.

10.3.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters Timer A and Timer B are initialized to all 1s, along with their corresponding registers:

- Load Registers:
  - **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 688)
  - **GPTM Timer B Interval Load (GPTMTBILR)** register (see page 689)
- Shadow Registers:
  - **GPTM Timer A Value (GPTMTAV)** register (see page 698)
  - **GPTM Timer B Value (GPTMTBV)** register (see page 699)

The following prescale counters are initialized to all 0s:

- **GPTM Timer A Prescale (GPTMTAPR)** register (see page 692)
- **GPTM Timer B Prescale (GPTMTBPR)** register (see page 693)

- **GPTM Timer A Prescale Snapshot (GPTMTAPS)** register (see page 701)
- **GPTM Timer B Prescale Snapshot (GPTMTBPS)** register (see page 702)
- **GPTM Timer A Prescale Value (GPTMTAPV)** register (see page 703)
- **GPTM Timer B Prescale Value (GPTMTBPV)** register (see page 704)

10.3.2 Timer Modes

This section describes the operation of the various timer modes. When using Timer A and Timer B in concatenated mode, only the Timer A control and status bits must be used; there is no need to use Timer B control and status bits. The GPTM is placed into individual/split mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 659). In the following sections, the variable "n" is used in bit field and register names to imply either a Timer A function or a Timer B function. Throughout this section, the timeout event in down-count mode is 0x0 and in up-count mode is the value in the **GPTM Timer n Interval Load (GPTMTnILR)** and the optional **GPTM Timer n Prescale (GPTMTnPR)** registers, with the exception of RTC mode.

10.3.2.1 One-Shot/Periodic Timer Mode

The selection of one-shot or periodic mode is determined by the value written to the TnMR field of the **GPTM Timer n Mode (GPTMTnMR)** register (see page 661). The timer is configured to count up or down using the TnCDIR bit in the **GPTMTnMR** register.

When software sets the TnEN bit in the **GPTM Control (GPTMCTL)** register (see page 669), the timer begins counting up from 0x0 or down from its preloaded value. Alternatively, if the TnWOT bit is set in the **GPTMTnMR** register, once the TnEN bit is set, the timer waits for a trigger to begin counting (see "Wait-for-Trigger Mode" on page 650). Table 10-4 on page 641 shows the values that are loaded into the timer registers when the timer is enabled.

Table 10-4. Counter Values When the Timer is Enabled in Periodic or One-Shot Modes

Register	Count Down Mode	Count Up Mode
GPTMTnR	**GPTMTnILR**	0x0
GPTMTnV	**GPTMTnILR** in concatenated mode; **GPTMTnPR** in combination with **GPTMTnILR** in individual mode	0x0
GPTMTnPS	**GPTMTnPR** in individual mode; not available in concatenated mode	0x0 in individual mode; not available in concatenated mode
GPTMTnPV	**GPTMTnPR** in individual mode; not available in concatenated mode	0x0 in individual mode; not available in concatenated mode

When the timer is counting down and it reaches the timeout event (0x0), the timer reloads its start value from the **GPTMTnILR** and the **GPTMTnPR** registers on the next cycle. When the timer is counting up and it reaches the timeout event (the value in the **GPTMTnILR** and the optional **GPTMTnPR** registers), the timer reloads with 0x0. If configured to be a one-shot timer, the timer stops counting and clears the TnEN bit in the **GPTMCTL** register. If configured as a periodic timer, the timer starts counting again on the next cycle.

In periodic, snap-shot mode (TnMR field is 0x2 and the TnSNAPS bit is set in the **GPTMTnMR** register), the value of the timer at the time-out event is loaded into the **GPTMTnR** register and the value of the prescaler is loaded into the **GPTMTnPS** register. The free-running counter value is shown in the **GPTMTnV** register and the free-running prescaler value is shown in the **GPTMTnPV** register. In this manner, software can determine the time elapsed from the interrupt assertion to the

ISR entry by examining the snapshot values and the current value of the free-running timer. Snapshot mode is not available when the timer is configured in one-shot mode.

In addition to reloading the count value, the GPTM can generate interrupts, CCP outputs and triggers when it reaches the time-out event. The GPTM sets the TnTORIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 680), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 686). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register (see page 677), the GPTM also sets the TnTOMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 683).

By setting the TnMIE bit in the **GPTMTnMR** register, an interrupt condition can also be generated when the Timer value equals the value loaded into the **GPTM Timer n Match (GPTMTnMATCHR)** and **GPTM Timer n Prescale Match (GPTMTnPMR)** registers. This interrupt has the same status, masking, and clearing functions as the time-out interrupt, but uses the match interrupt bits instead (for example, the raw interrupt status is monitored via TnMRIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register). Note that the interrupt status bits are not updated by the hardware unless the TnMIE bit in the **GPTMTnMR** register is set, which is different than the behavior for the time-out interrupt. The ADC trigger is enabled by setting the TnOTE bit in **GPTMCTL**. If the ADC trigger is enabled, only a one-shot or periodic time-out event can produce an ADC trigger assertion. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 521.

If software updates the **GPTMTnILR** or the **GPTMTnPR** register while the counter is counting down, the counter loads the new value on the next clock cycle and continues counting from the new value if the TnILD bit in the **GPTMTnMR** register is clear. If the TnILD bit is set, the counter loads the new value after the next timeout. If software updates the **GPTMTnILR** or the **GPTMTnPR** register while the counter is counting up, the timeout event is changed on the next cycle to the new value. If software updates the **GPTM Timer n Value (GPTMTnV)** register while the counter is counting up or down, the counter loads the new value on the next clock cycle and continues counting from the new value. If software updates the **GPTMTnMATCHR** or the **GPTMTnPMR** registers, the new values are reflected on the next clock cycle if the TnMRSU bit in the **GPTMTnMR** register is clear. If the TnMRSU bit is set, the new value will not take effect until the next timeout.

When using a 32/64-bit wide timer block in a 64-bit mode, certain registers must be accessed in the manner described in “Accessing Concatenated 32/64-Bit Wide GPTM Register Values” on page 652.

If the TnSTALL bit in the **GPTMCTL** register is set and the RTCEN bit is not set in the **GPTMCTL** register, the timer freezes counting while the processor is halted by the debugger. The timer resumes counting when the processor resumes execution. If the RTCEN bit is set, it prevents the TnSTALL bit from freezing the count when the processor is halted by the debugger.

The following table shows a variety of configurations for a 16-bit free-running timer while using the prescaler. All values assume an 80-MHz clock with Tc=12.5 ns (clock period). The prescaler can only be used when a 16/32-bit timer is configured in 16-bit mode and when a 32/64-bit timer is configured in 32-bit mode.

Table 10-5. 16-Bit Timer With Prescaler Configurations

Prescale (8-bit value)	# of Timer Clocks (Tc)[a]	Max Time	Units
00000000	1	0.8192	ms
00000001	2	1.6384	ms
00000010	3	2.4576	ms
------------	--	--	--
11111101	254	208.0768	ms

Table 10-5. 16-Bit Timer With Prescaler Configurations *(continued)*

Prescale (8-bit value)	# of Timer Clocks (Tc)[a]	Max Time	Units
11111110	255	208.896	ms
11111111	256	209.7152	ms

a. Tc is the clock period.

The following table shows a variety of configurations for a 32-bit free-running timer using the prescaler while configured in 32/64-bit mode. All values assume an 80-MHz clock with Tc=12.5 ns (clock period).

Table 10-6. 32-Bit Timer (configured in 32/64-bit mode) With Prescaler Configurations

Prescale (16-bit value)	# of Timer Clocks (Tc)[a]	Max Time	Units
0x0000	1	53.687	s
0x0001	2	107.374	s
0x0002	3	214.748	s
------------	--	--	--
0xFFFD	65534	0.879	10^6 s
0xFFFE	65535	1.759	10^6 s
0xFFFF	65536	3.518	10^6 s

a. Tc is the clock period.

10.3.2.2 Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the Timer A and Timer B registers are configured as an up-counter. When RTC mode is selected for the first time after reset, the counter is loaded with a value of 0x1. All subsequent load values must be written to the **GPTM Timer n Interval Load (GPTMTnILR)** registers (see page 688). If the **GPTMTnILR** register is loaded with a new value, the counter begins counting at that value and rolls over at the fixed value of 0xFFFFFFFF. Table 10-7 on page 643 shows the values that are loaded into the timer registers when the timer is enabled.

Table 10-7. Counter Values When the Timer is Enabled in RTC Mode

Register	Count Down Mode	Count Up Mode
GPTMTnR	Not available	0x1
GPTMTnV	Not available	0x1
GPTMTnPS	Not available	Not available
GPTMTnPV	Not available	Not available

The input clock on a CCP0 input is required to be 32.768 KHz in RTC mode. The clock signal is then divided down to a 1-Hz rate and is passed along to the input of the counter.

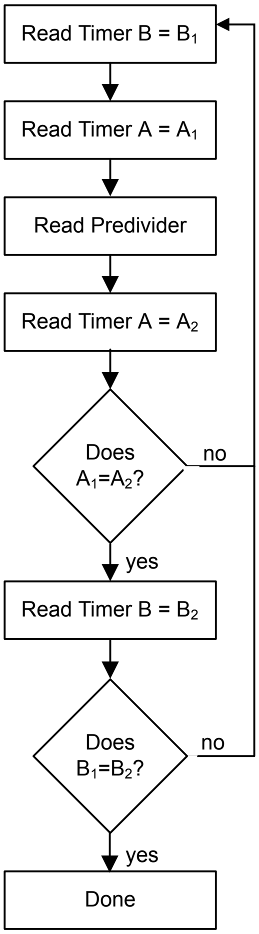
When software writes the TAEN bit in the **GPTMCTL** register, the counter starts counting up from its preloaded value of 0x1. When the current count value matches the preloaded value in the **GPTMTnMATCHR** registers, the GPTM asserts the RTCRIS bit in **GPTMRIS** and continues counting until either a hardware reset, or it is disabled by software (clearing the TAEN bit). When the timer value reaches the terminal count, the timer rolls over and continues counting up from 0x0. If the RTC interrupt is enabled in **GPTMIMR**, the GPTM also sets the RTCMIS bit in **GPTMMIS** and generates a controller interrupt. The status flags are cleared by writing the RTCCINT bit in **GPTMICR**.

In this mode, the **GPTMTnR** and **GPTMTnV** registers always have the same value.

When using a 32/64-bit wide timer block in a RTC mode, certain registers must be accessed in the manner described in “Accessing Concatenated 32/64-Bit Wide GPTM Register Values” on page 652.

The value of the RTC predivider can be read in the **GPTM RTC Predivide (GPTMRTCPD)** register. To ensure that the RTC value is coherent, software should follow the process detailed in Figure 10-2 on page 644.

Figure 10-2. Reading the RTC Value



In addition to generating interrupts, the RTC can generate a μDMA trigger. The μDMA trigger is enabled by configuring and enabling the appropriate μDMA channel. See “Channel Configuration” on page 521.

10.3.2.3 Input Edge-Count Mode

Note: For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

In Edge-Count mode, the timer is configured as a 24-bit or 48-bit up- or up- or down-counter including the optional prescaler with the upper count value stored in the **GPTM Timer n Prescale (GPTMTnPR)** register and the lower bits in the **GPTMTnR** register. In this mode, the timer is capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge-Count mode, the TnCMR bit of the **GPTMTnMR** register must be cleared. The type of edge that the timer counts is determined by the TnEVENT fields of the **GPTMCTL** register. During initialization in down-count mode, the **GPTMTnMATCHR** and **GPTMTnPMR** registers are configured so that the difference between the value in the **GPTMTnILR** and **GPTMTnPR** registers and the **GPTMTnMATCHR** and **GPTMTnPMR** registers equals the number of edge events that must be counted. In up-count mode, the timer counts from 0x0 to the value in the **GPTMTnMATCHR** and **GPTMTnPMR** registers. Note that when executing an up-count, that the value of **GPTMTnPR** and **GPTMTnILR** must be greater than the value of **GPTMTnPMR** and **GPTMTnMATCHR**. Table 10-8 on page 645 shows the values that are loaded into the timer registers when the timer is enabled.

Table 10-8. Counter Values When the Timer is Enabled in Input Edge-Count Mode

Register	Count Down Mode	Count Up Mode
GPTMTnR	**GPTMTnPR** in combination with **GPTMTnILR**	0x0
GPTMTnV	**GPTMTnPR** in combination with **GPTMTnILR**	0x0
GPTMTnPS	**GPTMTnPR**	0x0
GPTMTnPV	**GPTMTnPR**	0x0

When software writes the TnEN bit in the **GPTM Control (GPTMCTL)** register, the timer is enabled for event capture. Each input event on the CCP pin decrements or increments the counter by 1 until the event count matches **GPTMTnMATCHR** and **GPTMTnPMR**. When the counts match, the GPTM asserts the CnMRIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register, and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register. If the capture mode match interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register, the GPTM also sets the CnMMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register. In this mode, the **GPTMTnR** and **GPTMTnPS** registers hold the count of the input events while the **GPTMTnV** and **GPTMTnPV** registers hold the free-running timer value and the free-running prescaler value.In up count mode, the current count of input events is held in both the **GPTMTnR** and **GPTMTnV** registers.

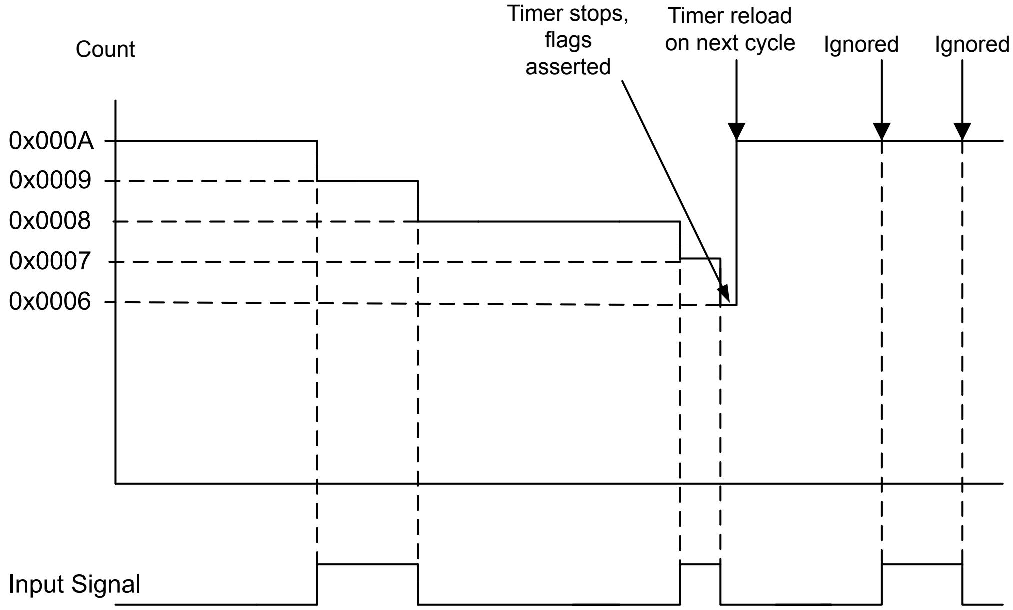
In addition to generating interrupts, a μDMA trigger can be generated. The μDMA trigger is enabled by configuring and enabling the appropriate μDMA channel. See “Channel Configuration” on page 521.

After the match value is reached in down-count mode, the counter is then reloaded using the value in **GPTMTnILR** and **GPTMTnPR** registers, and stopped because the GPTM automatically clears the TnEN bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until TnEN is re-enabled by software. In up-count mode, the timer is reloaded with 0x0 and continues counting.

Figure 10-3 on page 646 shows how Input Edge-Count mode works. In this case, the timer start value is set to **GPTMTnILR** =0x000A and the match value is set to **GPTMTnMATCHR** =0x0006 so that four edge events are counted. The counter is configured to detect both edges of the input signal.

Note that the last two edges are not counted because the timer automatically clears the TnEN bit after the current count matches the value in the **GPTMTnMATCHR** register.

Figure 10-3. Input Edge-Count Mode Example, Counting Down



10.3.2.4 Input Edge-Time Mode

Note: For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

In Edge-Time mode, the timer is configured as a 24-bit or 48-bit up- or down-counter including the optional prescaler with the upper timer value stored in the **GPTMTnPR** register and the lower bits in the **GPTMTnILR** register. In this mode, the timer is initialized to the value loaded in the **GPTMTnILR** and **GPTMTnPR** registers when counting down and 0x0 when counting up. The timer is capable of capturing three types of events: rising edge, falling edge, or both. The timer is placed into Edge-Time mode by setting the `TnCMR` bit in the **GPTMTnMR** register, and the type of event that the timer captures is determined by the `TnEVENT` fields of the **GPTMCTL** register. Table 10-9 on page 646 shows the values that are loaded into the timer registers when the timer is enabled.

Table 10-9. Counter Values When the Timer is Enabled in Input Event-Count Mode

Register	Count Down Mode	Count Up Mode
TnR	**GPTMTnILR**	0x0
TnV	**GPTMTnILR**	0x0
TnPS	**GPTMTnPR**	0x0
TnPV	**GPTMTnPR**	0x0

When software writes the `TnEN` bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current timer counter value is captured in the **GPTMTnR** and **GPTMTnPS** register and is available to be read by the microcontroller. The GPTM then asserts the `CnERIS` bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register, and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register. If the capture mode event interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register, the GPTM also sets the `CnEMIS` bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register. In this mode, the

GPTMTnR and **GPTMTnPS** registers hold the time at which the selected input event occurred while the **GPTMTnV** and **GPTMTnPV** registers hold the free-running timer value and the free-running prescaler value. These registers can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

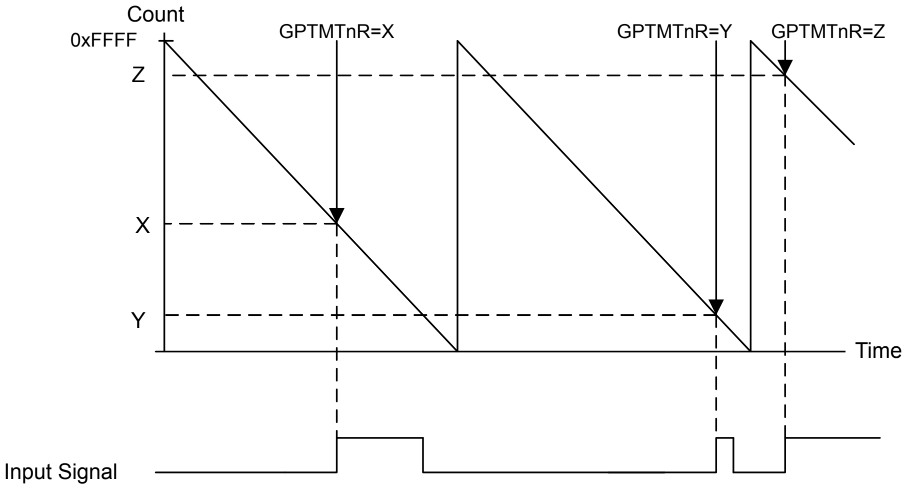
In addition to generating interrupts, a μDMA trigger can be generated. The μDMA trigger is enabled by configuring the appropriate μDMA channel. See "Channel Configuration" on page 521.

After an event has been captured, the timer does not stop counting. It continues to count until the TnEN bit is cleared. When the timer reaches the timeout value, it is reloaded with 0x0 in up-count mode and the value from the **GPTMTnILR** and **GPTMTnPR** registers in down-count mode.

Figure 10-4 on page 647 shows how input edge timing mode works. In the diagram, it is assumed that the start value of the timer is the default value of 0xFFFF, and the timer is configured to capture rising edge events.

Each time a rising edge event is detected, the current count value is loaded into the **GPTMTnR** and **GPTMTnPS** registers, and is held there until another rising edge is detected (at which point the new count value is loaded into the **GPTMTnR** and **GPTMTnPS** registers).

Figure 10-4. 16-Bit Input Edge-Time Mode Example



Note: When operating in Edge-time mode, the counter uses a modulo 2^{24} count if prescaler is enabled or 2^{16}, if not. If there is a possibility the edge could take longer than the count, then another timer configured in periodic-timer mode can be implemented to ensure detection of the missed edge. The periodic timer should be configured in such a way that:

- The periodic timer cycles at the same rate as the edge-time timer
- The periodic timer interrupt has a higher interrupt priority than the edge-time timeout interrupt.
- If the periodic timer interrupt service routine is entered, software must check if an edge-time interrupt is pending and if it is, the value of the counter must be subtracted by 1 before being used to calculate the snapshot time of the event.

10.3.2.5 PWM Mode

The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a 24-bit or 48-bit down-counter with a start value (and thus period) defined by the **GPTMTnILR** and **GPTMTnPR** registers. In this mode, the PWM frequency and period are synchronous events and therefore guaranteed to be glitch free. PWM mode is enabled with the **GPTMTnMR** register by setting the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2. Table 10-10 on page 648 shows the values that are loaded into the timer registers when the timer is enabled.

Table 10-10. Counter Values When the Timer is Enabled in PWM Mode

Register	Count Down Mode	Count Up Mode
GPTMTnR	**GPTMTnILR**	Not available
GPTMTnV	**GPTMTnILR**	Not available
GPTMTnPS	**GPTMTnPR**	Not available
GPTMTnPV	**GPTMTnPR**	Not available

When software writes the TnEN bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0 state. Alternatively, if the TnWOT bit is set in the **GPTMTnMR** register, once the TnEN bit is set, the timer waits for a trigger to begin counting (see “Wait-for-Trigger Mode” on page 650). On the next counter cycle in periodic mode, the counter reloads its start value from the **GPTMTnILR** and **GPTMTnPR** registers and continues counting until disabled by software clearing the TnEN bit in the **GPTMCTL** register. The timer is capable of generating interrupts based on three types of events: rising edge, falling edge, or both. The event is configured by the TnEVENT field of the **GPTMCTL** register, and the interrupt is enabled by setting the TnPWMIE bit in the **GPTMTnMR** register. When the event occurs, the CnERIS bit is set in the **GPTM Raw Interrupt Status (GPTMRIS)** register, and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register . If the capture mode event interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register , the GPTM also sets the CnEMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register. Note that the interrupt status bits are not updated unless the TnPWMIE bit is set.

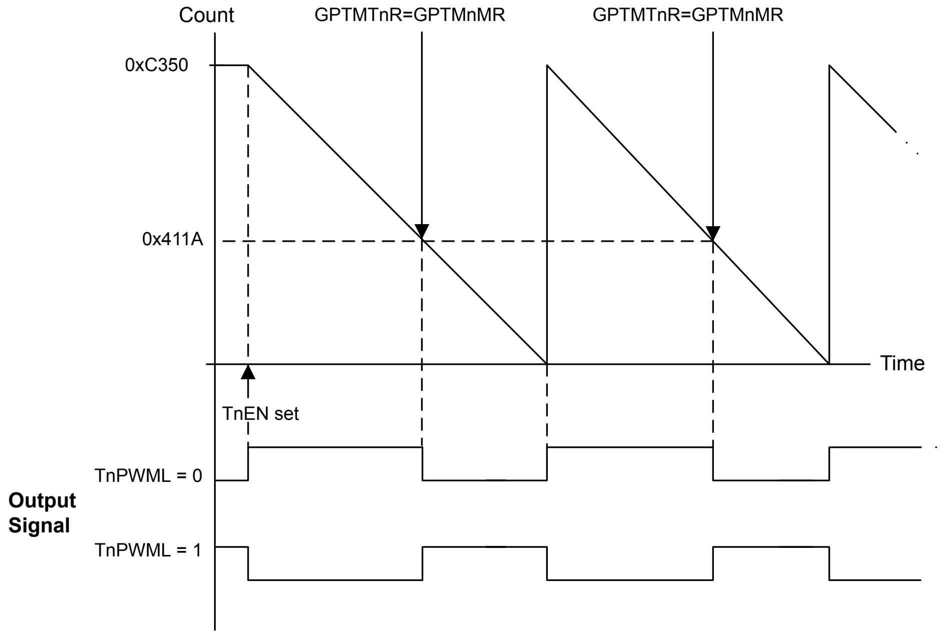
In this mode, the **GPTMTnR** and **GPTMTnV** registers always have the same value, as do the **GPTMPnPS** and the **GPTMTnPV** registers.

The output PWM signal asserts when the counter is at the value of the **GPTMTnILR** and **GPTMTnPR** registers (its start state), and is deasserted when the counter value equals the value in the **GPTMTnMATCHR** and **GPTMTnPMR** registers. Software has the capability of inverting the output PWM signal by setting the TnPWML bit in the **GPTMCTL** register.

Note: If PWM output inversion is enabled, edge detection interrupt behavior is reversed. Thus, if a positive-edge interrupt trigger has been set and the PWM inversion generates a positive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal.

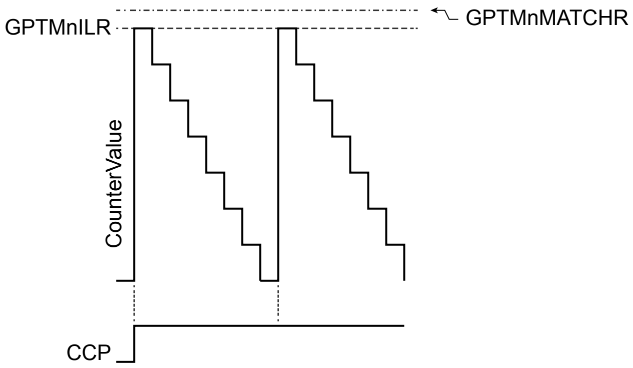
Figure 10-5 on page 649 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and **TnPWML** =0 (duty cycle would be 33% for the **TnPWML** =1 configuration). For this example, the start value is **GPTMTnILR**=0xC350 and the match value is **GPTMTnMATCHR**=0x411A.

Figure 10-5. 16-Bit PWM Mode Example



When synchronizing the timers using the **GPTMSYNC** register, the timer must be properly configured to avoid glitches on the CCP outputs. Both the `TnPLO` and the `TnMRSU` bits must be set in the **GPTMTnMR** register. Figure 10-6 on page 649 shows how the CCP output operates when the `TnPLO` and `TnMRSU` bits are set and the **GPTMTnMATCHR** value is greater than the **GPTMTnILR** value.

Figure 10-6. CCP Output, GPTMTnMATCHR > GPTMTnILR



CCP set if GPTMnMATCHR ≠ GPTMnILR

Figure 10-7 on page 650 shows how the CCP output operates when the `PLO` and `MRSU` bits are set and the **GPTMTnMATCHR** value is the same as the **GPTMTnILR** value. In this situation, if the **PLO** bit is 0, the CCP signal goes high when the **GPTMTnILR** value is loaded and the match would be essentially ignored.

Figure 10-7. CCP Output, GPTMTnMATCHR = GPTMTnILR

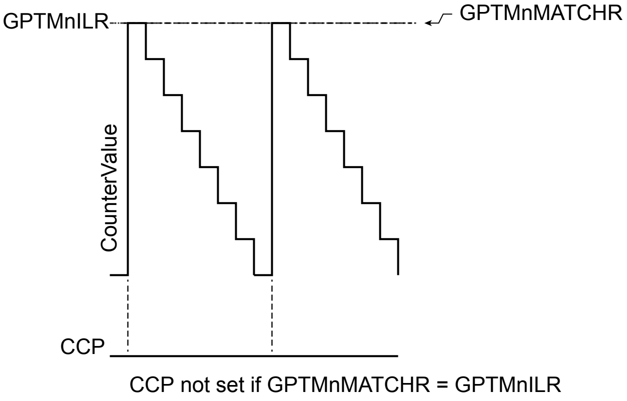
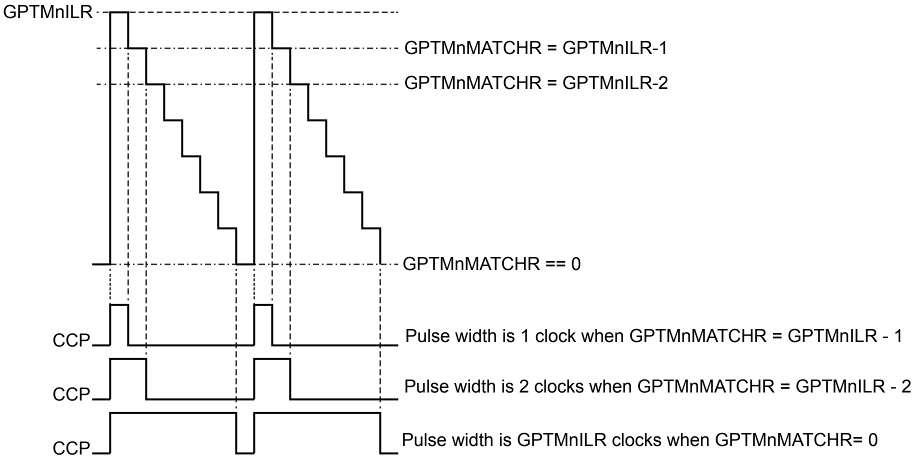


Figure 10-8 on page 650 shows how the CCP output operates when the `PLO` and `MRSU` bits are set and the **GPTMTnILR** is greater than the **GPTMTnMATCHR** value.

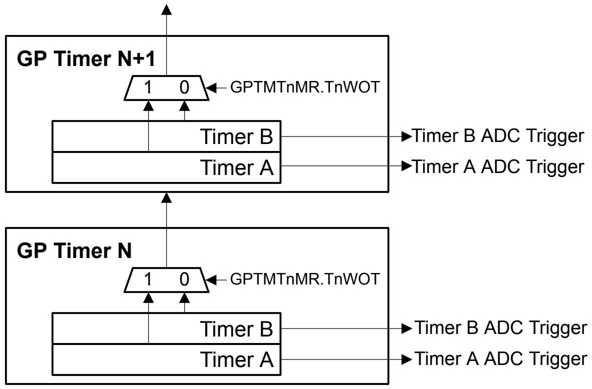
Figure 10-8. CCP Output, GPTMTnILR > GPTMTnMATCHR


GPTMnILR
GPTMnMATCHR = GPTMnILR-1
GPTMnMATCHR = GPTMnILR-2
GPTMnMATCHR == 0
CCP
Pulse width is 1 clock when GPTMnMATCHR = GPTMnILR - 1
CCP
Pulse width is 2 clocks when GPTMnMATCHR = GPTMnILR - 2
CCP
Pulse width is GPTMnILR clocks when GPTMnMATCHR= 0

10.3.3 Wait-for-Trigger Mode

The Wait-for-Trigger mode allows daisy chaining of the timer modules such that once configured, a single timer can initiate multiple timing events using the Timer triggers. Wait-for-Trigger mode is enabled by setting the `TnWOT` bit in the **GPTMTnMR** register. When the `TnWOT` bit is set, Timer N+1 does not begin counting until the timer in the previous position in the daisy chain (Timer N) reaches its time-out event. The daisy chain is configured such that GPTM1 always follows GPTM0, GPTM2 follows GPTM1, and so on. If Timer A is configured as a 32-bit (16/32-bit mode) or 64-bit (32/64-bit wide mode) timer (controlled by the `GPTMCFG` field in the **GPTMCFG** register), it triggers Timer A in the next module. If Timer A is configured as a 16-bit (16/32-bit mode) or 32-bit (32/64-bit wide mode) timer, it triggers Timer B in the same module, and Timer B triggers Timer A in the next module. Care must be taken that the `TAWOT` bit is never set in GPTM0. Figure 10-9 on page 651 shows how the `GPTMCFG` bit affects the daisy chain. This function is valid for one-shot, periodic, and PWM modes.

Figure 10-9. Timer Daisy Chain



10.3.4 Synchronizing GP Timer Blocks

The **GPTM Synchronizer Control (GPTMSYNC)** register in the GPTM0 block can be used to synchronize selected timers to begin counting at the same time. Setting a bit in the **GPTMSYNC** register causes the associated timer to perform the actions of a timeout event. An interrupt is not generated when the timers are synchronized. If a timer is being used in concatenated mode, only the bit for Timer A must be set in the **GPTMSYNC** register.

Note: All timers must use the same clock source for this feature to work correctly.

Table 10-11 on page 651 shows the actions for the timeout event performed when the timers are synchronized in the various timer modes.

Table 10-11. Timeout Actions for GPTM Modes

Mode	Count Dir	Time Out Action
32- and 64-bit One-Shot (concatenated timers)	—	N/A
32- and 64-bit Periodic (concatenated timers)	Down	Count value = ILR
	Up	Count value = 0
32- and 64-bit RTC (concatenated timers)	Up	Count value = 0
16- and 32- bit One Shot (individual/split timers)	—	N/A
16- and 32- bit Periodic (individual/split timers)	Down	Count value = ILR
	Up	Count value = 0
16- and 32- bit Edge-Count (individual/split timers)	Down	Count value = ILR
	Up	Count value = 0
16- and 32- bit Edge-Time (individual/split timers)	Down	Count value = ILR
	Up	Count value = 0
16- and 32-bit PWM	Down	Count value = ILR

10.3.5 DMA Operation

The timers each have a dedicated μDMA channel and can provide a request signal to the μDMA controller. The request is a burst type and occurs whenever a timer raw interrupt condition occurs. The arbitration size of the μDMA transfer should be set to the amount of data that should be transferred whenever a timer event occurs.

For example, to transfer 256 items, 8 items at a time every 10 ms, configure a timer to generate a periodic timeout at 10 ms. Configure the μDMA transfer for a total of 256 items, with a burst size of 8 items. Each time the timer times out, the μDMA controller transfers 8 items, until all 256 items have been transferred.

No other special steps are needed to enable Timers for μDMA operation. Refer to "Micro Direct Memory Access (μDMA)" on page 517 for more details about programming the μDMA controller.

10.3.6 Accessing Concatenated 16/32-Bit GPTM Register Values

The GPTM is placed into concatenated mode by writing a 0x0 or a 0x1 to the `GPTMCFG` bit field in the **GPTM Configuration (GPTMCFG)** register. In both configurations, certain 16/32-bit GPTM registers are concatenated to form pseudo 32-bit registers. These registers include:

- **GPTM Timer A Interval Load (GPTMTAILR)** register [15:0], see page 688
- **GPTM Timer B Interval Load (GPTMTBILR)** register [15:0], see page 689
- **GPTM Timer A (GPTMTAR)** register [15:0], see page 696
- **GPTM Timer B (GPTMTBR)** register [15:0], see page 697
- **GPTM Timer A Value (GPTMTAV)** register [15:0], see page 698
- **GPTM Timer B Value (GPTMTBV)** register [15:0], see page 699
- **GPTM Timer A Match (GPTMTAMATCHR)** register [15:0], see page 690
- **GPTM Timer B Match (GPTMTBMATCHR)** register [15:0], see page 691

In the 32-bit modes, the GPTM translates a 32-bit write access to **GPTMTAILR** into a write access to both **GPTMTAILR** and **GPTMTBILR**. The resulting word ordering for such a write operation is:

```
GPTMTBILR[15:0]:GPTMTAILR[15:0]
```

Likewise, a 32-bit read access to **GPTMTAR** returns the value:

```
GPTMTBR[15:0]:GPTMTAR[15:0]
```

A 32-bit read access to **GPTMTAV** returns the value:

```
GPTMTBV[15:0]:GPTMTAV[15:0]
```

10.3.7 Accessing Concatenated 32/64-Bit Wide GPTM Register Values

On the 32/64-bit wide GPTM blocks, concatenated register values (64-bits and 48-bits) are not readily available as the bit width for these accesses is greater than the bus width of the processor core. In the concatenated timer modes and the individual timer modes when using the prescaler, software must perform atomic accesses for the value to be coherent. When reading timer values that are greater than 32 bits, software should follow these steps:

1. Read the appropriate Timer B register or prescaler register.
2. Read the corresponding Timer A register.
3. Re-read the Timer B register or prescaler register.
4. Compare the Timer B or prescaler values from the first and second reads. If they are the same, the timer value is coherent. If they are not the same, repeat steps 1-4 once more so that they are the same.

The following pseudo code illustrates this process:

```
high = timer_high;

low = timer_low;

if (high != timer_high);  //low overflowed into high

{

    high = timer_high;

    low = timer_low;

}
```

The registers that must be read in this manner are shown below:

- 64-bit reads
  - **GPTMTAV** and **GPTMTBV**
  - **GPTMTAR** and **GPTMTBR**
- 48-bit reads
  - **GPTMTAR** and **GPTMTAPS**
  - **GPTMTBR** and **GPTMTBPS**
  - **GPTMTAV** and **GPTMTAPV**
  - **GPTMTBV** and **GPTMTBPV**

Similarly, write accesses must also be performed by writing the upper bits prior to writing the lower bits as follows:

1. Write the appropriate Timer B register or prescaler register.
2. Write the corresponding Timer A register.

The registers that must be written in this manner are shown below:

- 64-bit writes
  - **GPTMTAV** and **GPTMTBV**

  - **GPTMTAMATCHR** and **GPTMTBMATCHR**
  - **GPTMTAILR** and **GPTMTBILR**
- 48-bit writes
  - **GPTMTAV** and **GPTMTAPV**
  - **GPTMTBV** and **GPTMTBPV**
  - **GPTMTAMATCHR** and **GPTMTAPMR**
  - **GPTMTBMATCHR** and **GPTMTBPMR**
  - **GPTMTAILR** and **GPTMTAPR**
  - **GPTMTBILR** and **GPTMTBPR**

When writing a 64-bit value, If there are two consecutive writes to any of the registers listed above under the "64-bit writes" heading, whether the register is in Timer A or Timer B, or if a register Timer A is written prior to writing the corresponding register in Timer B, then an error is reported using the WUERIS bit in the **GPTMRIS** register. This error can be promoted to interrupt if it is not masked. Note that this error is not reported for the prescaler registers because use of the prescaler is optional. As a result, programmers must take care to follow the protocol outlined above.

10.4 Initialization and Configuration

To use a GPTM, the appropriate TIMERn bit must be set in the **RCGCTIMER** or **RCGCWTIMER** register (see page 317 and page 333). If using any CCP pins, the clock to the appropriate GPIO module must be enabled via the **RCGCGPIO** register (see page 319). To find out which GPIO port to enable, refer to Table 20-4 on page 1111. Configure the PMCn fields in the **GPIOPCTL** register to assign the CCP signals to the appropriate pins (see page 620 and Table 20-5 on page 1115).

This section shows module initialization and configuration examples for each of the supported timer modes.

10.4.1 One-Shot/Periodic Timer Mode

The GPTM is configured for One-Shot and Periodic modes by the following sequence:

1. Ensure the timer is disabled (the TnEN bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0000.
3. Configure the TnMR field in the **GPTM Timer n Mode Register (GPTMTnMR)**:
   a. Write a value of 0x1 for One-Shot mode.
   b. Write a value of 0x2 for Periodic mode.
4. Optionally configure the TnSNAPS, TnWOT, TnMTE, and TnCDIR bits in the **GPTMTnMR** register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.
5. Load the start value into the **GPTM Timer n Interval Load Register (GPTMTnILR)**.

6. If interrupts are required, set the appropriate bits in the **GPTM Interrupt Mask Register (GPTMIMR)**.

7. Set the TnEN bit in the **GPTMCTL** register to enable the timer and start counting.

8. Poll the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the appropriate bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

If the TnMIE bit in the **GPTMTnMR** register is set, the RTCRIS bit in the **GPTMRIS** register is set, and the timer continues counting. In One-Shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode reloads the timer and continues counting after the time-out event.

10.4.2 Real-Time Clock (RTC) Mode

To use the RTC mode, the timer must have a 32.768-KHz input signal on an even CCP input. To enable the RTC feature, follow these steps:

1. Ensure the timer is disabled (the TAEN bit is cleared) before making any changes.

2. If the timer has been operating in a different mode prior to this, clear any residual set bits in the **GPTM Timer n Mode (GPTMTnMR)** register before reconfiguring.

3. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0001.

4. Write the match value to the **GPTM Timer n Match Register (GPTMTnMATCHR)**.

5. Set/clear the RTCEN and TnSTALL bit in the **GPTM Control Register (GPTMCTL)** as needed.

6. If interrupts are required, set the RTCIM bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.

7. Set the TAEN bit in the **GPTMCTL** register to enable the timer and start counting.

When the timer count equals the value in the **GPTMTnMATCHR** register, the GPTM asserts the RTCRIS bit in the **GPTMRIS** register and continues counting until Timer A is disabled or a hardware reset. The interrupt is cleared by writing the RTCCINT bit in the **GPTMICR** register. Note that if the **GPTMTnILR** register is loaded with a new value, the timer begins counting at this new value and continues until it reaches 0xFFFF.FFFF, at which point it rolls over.

10.4.3 Input Edge-Count Mode

A timer is configured to Input Edge-Count mode by the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.

2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.

3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the TnCMR field to 0x0 and the TnMR field to 0x3.

4. Configure the type of event(s) that the timer captures by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.

5. Program registers according to count direction:

- In down-count mode, the **GPTMTnMATCHR** and **GPTMTnPMR** registers are configured so that the difference between the value in the **GPTMTnILR** and **GPTMTnPR** registers and the **GPTMTnMATCHR** and **GPTMTnPMR** registers equals the number of edge events that must be counted.

- In up-count mode, the timer counts from 0x0 to the value in the **GPTMTnMATCHR** and **GPTMTnPMR** registers. Note that when executing an up-count, the value of the **GPTMTnPR** and **GPTMTnILR** must be greater than the value of **GPTMTnPMR** and **GPTMTnMATCHR**.

6. If interrupts are required, set the `CnMIM` bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
7. Set the `TnEN` bit in the **GPTMCTL** register to enable the timer and begin waiting for edge events.
8. Poll the `CnMRIS` bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the `CnMCINT` bit of the **GPTM Interrupt Clear (GPTMICR)** register.

When counting down in Input Edge-Count Mode, the timer stops after the programmed number of edge events has been detected. To re-enable the timer, ensure that the `TnEN` bit is cleared and repeat steps 4 through 8.

10.4.4 Input Edge Time Mode

A timer is configured to Input Edge Time mode by the following sequence:

1. Ensure the timer is disabled (the `TnEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the `TnCMR` field to 0x1 and the `TnMR` field to 0x3 and select a count direction by programming the `TnCDIR` bit.
4. Configure the type of event that the timer captures by writing the `TnEVENT` field of the **GPTM Control (GPTMCTL)** register.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
7. If interrupts are required, set the `CnEIM` bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
8. Set the `TnEN` bit in the **GPTM Control (GPTMCTL)** register to enable the timer and start counting.
9. Poll the `CnERIS` bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the `CnECINT` bit of the **GPTM Interrupt Clear (GPTMICR)** register. The time at which the event happened can be obtained by reading the **GPTM Timer n (GPTMTnR)** register.

In Input Edge Timing mode, the timer continues running after an edge event has been detected, but the timer interval can be changed at any time by writing the **GPTMTnILR** register and clearing the `TnILD` bit in the **GPTMTnMR** register. The change takes effect at the next cycle after the write.

10.4.5 PWM Mode

A timer is configured to PWM mode using the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, set the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the TnPWML field of the **GPTM Control (GPTMCTL)** register.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. If PWM interrupts are used, configure the interrupt condition in the TnEVENT field in the **GPTMCTL** register and enable the interrupts by setting the TnPWMIE bit in the **GPTMTnMR** register. Note that edge detect interrupt behavior is reversed when the PWM output is inverted (see page 669).
7. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
8. Load the **GPTM Timer n Match (GPTMTnMATCHR)** register with the match value.
9. Set the TnEN bit in the **GPTM Control (GPTMCTL)** register to enable the timer and begin generation of the output PWM signal.

In PWM Time mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing the **GPTMTnILR** register, and the change takes effect at the next cycle after the write.

10.5 Register Map

Table 10-12 on page 658 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

- 16/32-bit Timer 0: 0x4003.0000
- 16/32-bit Timer 1: 0x4003.1000
- 16/32-bit Timer 2: 0x4003.2000
- 16/32-bit Timer 3: 0x4003.3000
- 16/32-bit Timer 4: 0x4003.4000
- 16/32-bit Timer 5: 0x4003.5000
- 32/64-bit Wide Timer 0: 0x4003.6000
- 32/64-bit Wide Timer 1: 0x4003.7000
- 32/64-bit Wide Timer 2: 0x4004.C000
- 32/64-bit Wide Timer 3: 0x4004.D000
- 32/64-bit Wide Timer 4: 0x4004.E000
- 32/64-bit Wide Timer 5: 0x4004.F000

The SIZE field in the **GPTM Peripheral Properties (GPTMPP)** register identifies whether a module has a 16/32-bit or 32/64-bit wide timer.

Note that the GP Timer module clock must be enabled before the registers can be programmed (see page 317 or page 333). There must be a delay of 3 system clocks after the Timer module clock is enabled before any Timer module registers are accessed.

Table 10-12. Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	RW	0x0000.0000	GPTM Configuration	659
0x004	GPTMTAMR	RW	0x0000.0000	GPTM Timer A Mode	661
0x008	GPTMTBMR	RW	0x0000.0000	GPTM Timer B Mode	665
0x00C	GPTMCTL	RW	0x0000.0000	GPTM Control	669
0x010	GPTMSYNC	RW	0x0000.0000	GPTM Synchronize	673
0x018	GPTMIMR	RW	0x0000.0000	GPTM Interrupt Mask	677
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	680
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	683
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	686
0x028	GPTMTAILR	RW	0xFFFF.FFFF	GPTM Timer A Interval Load	688
0x02C	GPTMTBILR	RW	-	GPTM Timer B Interval Load	689
0x030	GPTMTAMATCHR	RW	0xFFFF.FFFF	GPTM Timer A Match	690
0x034	GPTMTBMATCHR	RW	-	GPTM Timer B Match	691
0x038	GPTMTAPR	RW	0x0000.0000	GPTM Timer A Prescale	692
0x03C	GPTMTBPR	RW	0x0000.0000	GPTM Timer B Prescale	693
0x040	GPTMTAPMR	RW	0x0000.0000	GPTM TimerA Prescale Match	694
0x044	GPTMTBPMR	RW	0x0000.0000	GPTM TimerB Prescale Match	695
0x048	GPTMTAR	RO	0xFFFF.FFFF	GPTM Timer A	696
0x04C	GPTMTBR	RO	-	GPTM Timer B	697
0x050	GPTMTAV	RW	0xFFFF.FFFF	GPTM Timer A Value	698
0x054	GPTMTBV	RW	-	GPTM Timer B Value	699
0x058	GPTMRTCPD	RO	0x0000.7FFF	GPTM RTC Predivide	700
0x05C	GPTMTAPS	RO	0x0000.0000	GPTM Timer A Prescale Snapshot	701
0x060	GPTMTBPS	RO	0x0000.0000	GPTM Timer B Prescale Snapshot	702
0x064	GPTMTAPV	RO	0x0000.0000	GPTM Timer A Prescale Value	703
0x068	GPTMTBPV	RO	0x0000.0000	GPTM Timer B Prescale Value	704
0xFC0	GPTMPP	RO	0x0000.0000	GPTM Peripheral Properties	705

10.6 Register Descriptions

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

Register 1: GPTM Configuration (GPTMCFG), offset 0x000

This register configures the global operation of the GPTM module. The value written to this register determines whether the GPTM is in 32- or 64-bit mode (concatenated timers) or in 16- or 32-bit mode (individual, split timers).

Important: Bits in this register should only be changed when the TAEN and TBEN bits in the **GPTMCTL** register are cleared.

GPTM Configuration (GPTMCFG)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x000
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													GPTMCFG		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2:0	GPTMCFG	RW	0x0	GPTM Configuration The `GPTMCFG` values are defined as follows:

Value	Description
0x0	For a 16/32-bit timer, this value selects the 32-bit timer configuration. For a 32/64-bit wide timer, this value selects the 64-bit timer configuration.
0x1	For a 16/32-bit timer, this value selects the 32-bit real-time clock (RTC) counter configuration. For a 32/64-bit wide timer, this value selects the 64-bit real-time clock (RTC) counter configuration.
0x2-0x3	Reserved
0x4	For a 16/32-bit timer, this value selects the 16-bit timer configuration. For a 32/64-bit wide timer, this value selects the 32-bit timer configuration. The function is controlled by bits 1:0 of **GPTMTAMR** and **GPTMTBMR**.
0x5-0x7	Reserved

Register 2: GPTM Timer A Mode (GPTMTAMR), offset 0x004

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in PWM mode, set the `TAAMS` bit, clear the `TACMR` bit, and configure the `TAMR` field to 0x1 or 0x2.

This register controls the modes for Timer A when it is used individually. When Timer A and Timer B are concatenated, this register controls the modes for both Timer A and Timer B, and the contents of **GPTMTBMR** are ignored.

Important: Bits in this register should only be changed when the `TAEN` bit in the **GPTMCTL** register is cleared.

GPTM Timer A Mode (GPTMTAMR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x004
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TAPLO	TAMRSU	TAPWMIE	TAILD	TASNAPS	TAWOT	TAMIE	TACDIR	TAAMS	TACMR	TAMR	
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.00	Software should not rely on the value of 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	TAPLO	RW	0	GPTM Timer A PWM Legacy Operation Value Description 0 Legacy operation with CCP pin driven Low when the **GPTMTAILR** is reloaded after the timer reaches 0. 1 CCP is driven High when the **GPTMTAILR** is reloaded after the timer reaches 0. This bit is only valid in PWM mode.

Bit/Field	Name	Type	Reset	Description
10	TAMRSU	RW	0	GPTM Timer A Match Register Update Value Description 0 Update the **GPTMTAMATCHR** register and the **GPTMTAPR** register, if used, on the next cycle. 1 Update the **GPTMTAMATCHR** register and the **GPTMTAPR** register, if used, on the next timeout. If the timer is disabled (TAEN is clear) when this bit is set, **GPTMTAMATCHR** and **GPTMTAPR** are updated when the timer is enabled. If the timer is stalled (TASTALL is set), **GPTMTAMATCHR** and **GPTMTAPR** are updated according to the configuration of this bit.
9	TAPWMIE	RW	0	GPTM Timer A PWM Interrupt Enable This bit enables interrupts in PWM mode on rising, falling, or both edges of the CCP output, as defined by the TAEVENT field in the **GPTMCTL** register. Value Description 0 Capture event interrupt is disabled. 1 Capture event interrupt is enabled. This bit is only valid in PWM mode.
8	TAILD	RW	0	GPTM Timer A Interval Load Write Value Description 0 Update the **GPTMTAR** and **GPTMTAV** registers with the value in the **GPTMTAILR** register on the next cycle. Also update the **GPTMTAPS** and **GPTMTAPV** registers with the value in the **GPTMTAPR** register on the next cycle. 1 Update the **GPTMTAR** and **GPTMTAV** registers with the value in the **GPTMTAILR** register on the next timeout. Also update the **GPTMTAPS** and **GPTMTAPV** registers with the value in the **GPTMTAPR** register on the next timeout. Note the state of this bit has no effect when counting up. The bit descriptions above apply if the timer is enabled and running. If the timer is disabled (TAEN is clear) when this bit is set, **GPTMTAR**, **GPTMTAV** and **GPTMTAPs**, and **GPTMTAPV** are updated when the timer is enabled. If the timer is stalled (TASTALL is set), **GPTMTAR** and **GPTMTAPS** are updated according to the configuration of this bit.
7	TASNAPS	RW	0	GPTM Timer A Snap-Shot Mode Value Description 0 Snap-shot mode is disabled. 1 If Timer A is configured in the periodic mode, the actual free-running, capture or snapshot value of Timer A is loaded at the time-out event/capture or snapshot event into the **GPTM Timer A (GPTMTAR)** register. If the timer prescaler is used, the prescaler snapshot is loaded into the **GPTM Timer A (GPTMTAPR).**

Bit/Field	Name	Type	Reset	Description
6	TAWOT	RW	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 10-9 on page 651. This function is valid for one-shot, periodic, and PWM modes. This bit must be clear for GP Timer Module 0, Timer A.
5	TAMIE	RW	0	GPTM Timer A Match Interrupt Enable Value Description 0 The match interrupt is disabled for match events. **Note:** Clearing the TAMIE bit in the **GPTMTAMR** register prevents assertion of µDMA or ADC requests generated on a match event. Even if the TATODMAEN bit is set in the **GPTMDMAEV** register or the TATOADCEN bit is set in the **GPTMADCEV** register, a µDMA or ADC match trigger is not sent to the µDMA or ADC, respectively, when the TAMIE bit is clear. 1 An interrupt is generated when the match value in the **GPTMTAMATCHR** register is reached in the one-shot and periodic modes.
4	TACDIR	RW	0	GPTM Timer A Count Direction Value Description 0 The timer counts down. 1 The timer counts up. When counting up, the timer starts from a value of 0x0. When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up.
3	TAAMS	RW	0	GPTM Timer A Alternate Mode Select The TAAMS values are defined as follows: Value Description 0 Capture or compare mode is enabled. 1 PWM mode is enabled. **Note:** To enable PWM mode, you must also clear the TACMR bit and configure the TAMR field to 0x1 or 0x2.
2	TACMR	RW	0	GPTM Timer A Capture Mode The TACMR values are defined as follows: Value Description 0 Edge-Count mode 1 Edge-Time mode

Bit/Field	Name	Type	Reset	Description
1:0	TAMR	RW	0x0	GPTM Timer A Mode The TAMR values are defined as follows: Value Description 0x0 Reserved 0x1 One-Shot Timer mode 0x2 Periodic Timer mode 0x3 Capture mode The Timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register.

Register 3: GPTM Timer B Mode (GPTMTBMR), offset 0x008

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in PWM mode, set the TBAMS bit, clear the TBCMR bit, and configure the TBMR field to 0x1 or 0x2.

This register controls the modes for Timer B when it is used individually. When Timer A and Timer B are concatenated, this register is ignored and **GPTMTAMR** controls the modes for both Timer A and Timer B.

Important: Bits in this register should only be changed when the TBEN bit in the **GPTMCTL** register is cleared.

GPTM Timer B Mode (GPTMTBMR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x008
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TBPLO	TBMRSU	TBPWMIE	TBILD	TBSNAPS	TBWOT	TBMIE	TBCDIR	TBAMS	TBCMR	TBMR	
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.00	Software should not rely on the value of 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	TBPLO	RW	0	GPTM Timer B PWM Legacy Operation

Value	Description
0	Legacy operation with CCP pin driven Low when the **GPTMTAILR** is reloaded after the timer reaches 0.
1	CCP is driven High when the **GPTMTAILR** is reloaded after the timer reaches 0.

This bit is only valid in PWM mode.

Bit/Field	Name	Type	Reset	Description
10	TBMRSU	RW	0	GPTM Timer B Match Register Update Value Description 0 Update the **GPTMTBMATCHR** register and the **GPTMTBPR** register, if used, on the next cycle. 1 Update the **GPTMTBMATCHR** register and the **GPTMTBPR** register, if used, on the next timeout. If the timer is disabled (TBEN is clear) when this bit is set, **GPTMTBMATCHR** and **GPTMTBPR** are updated when the timer is enabled. If the timer is stalled (TBSTALL is set), **GPTMTBMATCHR** and **GPTMTBPR** are updated according to the configuration of this bit.
9	TBPWMIE	RW	0	GPTM Timer B PWM Interrupt Enable This bit enables interrupts in PWM mode on rising, falling, or both edges of the CCP output as defined by the TBEVENT field in the **GPTMCTL** register. Value Description 0 Capture event interrupt is disabled. 1 Capture event is enabled. This bit is only valid in PWM mode.
8	TBILD	RW	0	GPTM Timer B Interval Load Write Value Description 0 Update the **GPTMTBR** and **GPTMTBV** registers with the value in the **GPTMTBILR** register on the next cycle. Also update the **GPTMTBPS** and **GPTMTBPV** registers with the value in the **GPTMTBPR** register on the next cycle. 1 Update the **GPTMTBR** and **GPTMTBV** registers with the value in the **GPTMTBILR** register on the next timeout. Also update the **GPTMTBPS** and **GPTMTBPV** registers with the value in the **GPTMTBPR** register on the next timeout. Note the state of this bit has no effect when counting up. The bit descriptions above apply if the timer is enabled and running. If the timer is disabled (TBEN is clear) when this bit is set, **GPTMTBR, GPTMTBV** and, **GPTMTBPS**, and **GPTMTBPV** are updated when the timer is enabled. If the timer is stalled (TBSTALL is set), **GPTMTBR** and **GPTMTBPS** are updated according to the configuration of this bit.
7	TBSNAPS	RW	0	GPTM Timer B Snap-Shot Mode 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. If the timer prescaler is used, the prescaler snapshot is loaded into the **GPTM Timer B (GPTMTBPR).**

Bit/Field	Name	Type	Reset	Description
6	TBWOT	RW	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 a trigger from the timer in the previous position in the daisy chain, see Figure 10-9 on page 651. This function is valid for one-shot, periodic, and PWM modes.
5	TBMIE	RW	0	GPTM Timer B Match Interrupt Enable Value Description 0 The match interrupt is disabled for match events. 1 An interrupt is generated when the match value in the **GPTMTBMATCHR** register is reached in the one-shot and periodic modes. **Note:** Clearing the TBMIE bit in the **GPTMTBMR** register prevents assertion of µDMA or ADC requests generated on a match event. Even if the TBTODMAEN bit is set in the **GPTMDMAEV** register or the TBTOADCEN bit is set in the **GPTMADCEV** register, a µDMA or ADC match trigger is not sent to the µDMA or ADC, respectively, when the TBMIE bit is clear.
4	TBCDIR	RW	0	GPTM Timer B Count Direction Value Description 0 The timer counts down. 1 The timer counts up. When counting up, the timer starts from a value of 0x0. When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up.
3	TBAMS	RW	0	GPTM Timer B Alternate Mode Select The TBAMS values are defined as follows: Value Description 0 Capture or compare mode is enabled. 1 PWM mode is enabled. **Note:** To enable PWM mode, you must also clear the TBCMR bit and configure the TBMR field to 0x1 or 0x2.
2	TBCMR	RW	0	GPTM Timer B Capture Mode The TBCMR values are defined as follows: Value Description 0 Edge-Count mode 1 Edge-Time mode

Bit/Field	Name	Type	Reset	Description
1:0	TBMR	RW	0x0	GPTM Timer B Mode The `TBMR` values are defined as follows: Value Description 0x0 Reserved 0x1 One-Shot Timer mode 0x2 Periodic Timer mode 0x3 Capture mode The timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register.

Register 4: GPTM Control (GPTMCTL), offset 0x00C

This register is used alongside the **GPTMCFG** and **GMTMTnMR** registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger. The output trigger can be used to initiate transfers on the ADC module.

Important: Bits in this register should only be changed when the `TnEN` bit for the respective timer is cleared.

GPTM Control (GPTMCTL)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x00C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT		TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
Type	RO	RW	RW	RO	RW	RW	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	TBPWML	RW	0	GPTM Timer B PWM Output Level The `TBPWML` values are defined as follows:

Value	Description
0	Output is unaffected.
1	Output is inverted.

Bit/Field	Name	Type	Reset	Description
13	TBOTE	RW	0	GPTM Timer B Output Trigger Enable The `TBOTE` values are defined as follows: Value Description 0 The output Timer B ADC trigger is disabled. 1 The output Timer B ADC trigger is enabled. **Note:** The timer must be configured for one-shot or periodic time-out mode to produce an ADC trigger assertion. The GPTM does not generate triggers for match, compare events or compare match events. 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 765).
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	TBEVENT	RW	0x0	GPTM Timer B Event Mode The `TBEVENT` values are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges **Note:** If PWM output inversion is enabled, edge detection interrupt behavior is reversed. Thus, if a positive-edge interrupt trigger has been set and the PWM inversion generates a postive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal.
9	TBSTALL	RW	0	GPTM Timer B Stall Enable The `TBSTALL` values are defined as follows: Value Description 0 Timer B continues counting while the processor is halted by the debugger. 1 Timer B freezes counting while the processor is halted by the debugger. If the processor is executing normally, the `TBSTALL` bit is ignored.
8	TBEN	RW	0	GPTM Timer B Enable The `TBEN` values are defined as follows: 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.

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	TAPWML	RW	0	GPTM Timer A PWM Output Level The TAPWML values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
5	TAOTE	RW	0	GPTM Timer A Output Trigger Enable The TAOTE values are defined as follows: Value Description 0 The output Timer A ADC trigger is disabled. 1 The output Timer A ADC trigger is enabled. **Note:** The timer must be configured for one-shot or periodic time-out mode to produce an ADC trigger assertion. The GPTM does not generate triggers for match, compare events or compare match events. 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 765).
4	RTCEN	RW	0	GPTM RTC Stall Enable The RTCEN values are defined as follows: Value Description 0 RTC counting freezes while the processor is halted by the debugger. 1 RTC counting continues while the processor is halted by the debugger. If the RTCEN bit is set, it prevents the timer from stalling in all operating modes, even if TnSTALL is set.
3:2	TAEVENT	RW	0x0	GPTM Timer A Event Mode The TAEVENT values are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges **Note:** If PWM output inversion is enabled, edge detection interrupt behavior is reversed. Thus, if a positive-edge interrupt trigger has been set and the PWM inversion generates a postive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal.

Bit/Field	Name	Type	Reset	Description
1	TASTALL	RW	0	GPTM Timer A Stall Enable The TASTALL values are defined as follows: Value Description 0 Timer A continues counting while the processor is halted by the debugger. 1 Timer A freezes counting while the processor is halted by the debugger. If the processor is executing normally, the TASTALL bit is ignored.
0	TAEN	RW	0	GPTM Timer A Enable The TAEN values are defined as follows: Value Description 0 Timer A is disabled. 1 Timer A is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register.

Register 5: GPTM Synchronize (GPTMSYNC), offset 0x010

Note: This register is only implemented on GPTM Module 0 only.

This register allows software to synchronize a number of timers.

GPTM Synchronize (GPTMSYNC)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x010
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								SYNCWT5		SYNCWT4		SYNCWT3		SYNCWT2	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SYNCWT1		SYNCWT0		SYNCT5		SYNCT4		SYNCT3		SYNCT2		SYNCT1		SYNCT0	
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:22	SYNCWT5	RW	0x0	Synchronize GPTM 32/64-Bit Timer 5 The SYNCWT5 values are defined as follows:

Value	Description
0x0	GPTM 32/64-Bit Timer 5 is not affected.
0x1	A timeout event for Timer A of GPTM 32/64-Bit Timer 5 is triggered.
0x2	A timeout event for Timer B of GPTM 32/64-Bit Timer 5 is triggered.
0x3	A timeout event for both Timer A and Timer B of GPTM 32/64-Bit Timer 5 is triggered.

Bit/Field	Name	Type	Reset	Description
21:20	SYNCWT4	RW	0x0	Synchronize GPTM 32/64-Bit Timer 4 The `SYNCWT4` values are defined as follows: Value Description 0x0 GPTM 32/64-Bit Timer 4 is not affected. 0x1 A timeout event for Timer A of GPTM 32/64-Bit Timer 4 is triggered. 0x2 A timeout event for Timer B of GPTM 32/64-Bit Timer 4 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 32/64-Bit Timer 4 is triggered.
19:18	SYNCWT3	RW	0x0	Synchronize GPTM 32/64-Bit Timer 3 The `SYNCWT3` values are defined as follows: Value Description 0x0 GPTM 32/64-Bit Timer 3 is not affected. 0x1 A timeout event for Timer A of GPTM 32/64-Bit Timer 3 is triggered. 0x2 A timeout event for Timer B of GPTM 32/64-Bit Timer 3 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 32/64-Bit Timer 3 is triggered.
17:16	SYNCWT2	RW	0x0	Synchronize GPTM 32/64-Bit Timer 2 The `SYNCWT2` values are defined as follows: Value Description 0x0 GPTM 32/64-Bit Timer 2 is not affected. 0x1 A timeout event for Timer A of GPTM 32/64-Bit Timer 2 is triggered. 0x2 A timeout event for Timer B of GPTM 32/64-Bit Timer 2 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 32/64-Bit Timer 2 is triggered.
15:14	SYNCWT1	RW	0x0	Synchronize GPTM 32/64-Bit Timer 1 The `SYNCWT1` values are defined as follows: Value Description 0x0 GPTM 32/64-Bit Timer 1 is not affected. 0x1 A timeout event for Timer A of GPTM 32/64-Bit Timer 1 is triggered. 0x2 A timeout event for Timer B of GPTM 32/64-Bit Timer 1 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 32/64-Bit Timer 1 is triggered.

Bit/Field	Name	Type	Reset	Description
13:12	SYNCWT0	RW	0x0	Synchronize GPTM 32/64-Bit Timer 0 The `SYNCWT0` values are defined as follows: Value Description 0x0 GPTM 32/64-Bit Timer 0 is not affected. 0x1 A timeout event for Timer A of GPTM 32/64-Bit Timer 0 is triggered. 0x2 A timeout event for Timer B of GPTM 32/64-Bit Timer 0 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 32/64-Bit Timer 0 is triggered.
11:10	SYNCT5	RW	0x0	Synchronize GPTM 16/32-Bit Timer 5 The `SYNCT5` values are defined as follows: Value Description 0x0 GPTM 16/32-Bit Timer 5 is not affected. 0x1 A timeout event for Timer A of GPTM 16/32-Bit Timer 5 is triggered. 0x2 A timeout event for Timer B of GPTM 16/32-Bit Timer 5 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 16/32-Bit Timer 5 is triggered.
9:8	SYNCT4	RW	0x0	Synchronize GPTM 16/32-Bit Timer 4 The `SYNCT4` values are defined as follows: Value Description 0x0 GPTM 16/32-Bit Timer 4 is not affected. 0x1 A timeout event for Timer A of GPTM 16/32-Bit Timer 4 is triggered. 0x2 A timeout event for Timer B of GPTM 16/32-Bit Timer 4 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 16/32-Bit Timer 4 is triggered.
7:6	SYNCT3	RW	0x0	Synchronize GPTM 16/32-Bit Timer 3 The `SYNCT3` values are defined as follows: Value Description 0x0 GPTM 16/32-Bit Timer 3 is not affected. 0x1 A timeout event for Timer A of GPTM 16/32-Bit Timer 3 is triggered. 0x2 A timeout event for Timer B of GPTM 16/32-Bit Timer 3 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 16/32-Bit Timer 3 is triggered.

Bit/Field	Name	Type	Reset	Description
5:4	SYNCT2	RW	0x0	Synchronize GPTM 16/32-Bit Timer 2 The SYNCT2 values are defined as follows: Value Description 0x0 GPTM 16/32-Bit Timer 2 is not affected. 0x1 A timeout event for Timer A of GPTM 16/32-Bit Timer 2 is triggered. 0x2 A timeout event for Timer B of GPTM 16/32-Bit Timer 2 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 16/32-Bit Timer 2 is triggered.
3:2	SYNCT1	RW	0x0	Synchronize GPTM 16/32-Bit Timer 1 The SYNCT1 values are defined as follows: Value Description 0x0 GPTM 16/32-Bit Timer 1 is not affected. 0x1 A timeout event for Timer A of GPTM 16/32-Bit Timer 1 is triggered. 0x2 A timeout event for Timer B of GPTM 16/32-Bit Timer 1 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 16/32-Bit Timer 1 is triggered.
1:0	SYNCT0	RW	0x0	Synchronize GPTM 16/32-Bit Timer 0 The SYNCT0 values are defined as follows: Value Description 0x0 GPTM 16/32-Bit Timer 0 is not affected. 0x1 A timeout event for Timer A of GPTM 16/32-Bit Timer 0 is triggered. 0x2 A timeout event for Timer B of GPTM 16/32-Bit Timer 0 is triggered. 0x3 A timeout event for both Timer A and Timer B of GPTM 16/32-Bit Timer 0 is triggered.

Register 6: GPTM Interrupt Mask (GPTMIMR), offset 0x018

This register allows software to enable/disable GPTM controller-level interrupts. Setting a bit enables the corresponding interrupt, while clearing a bit disables it.

GPTM Interrupt Mask (GPTMIMR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x018
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															WUEIM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TBMIM	CBEIM	CBMIM	TBTOIM	reserved			TAMIM	RTCIM	CAEIM	CAMIM	TATOIM
Type	RO	RO	RO	RO	RW	RW	RW	RW	RO	RO	RO	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	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.
16	WUEIM	RW	0	32/64-Bit Wide GPTM Write Update Error Interrupt Mask The `WUEIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMIM	RW	0	GPTM Timer B Match Interrupt Mask The `TBMIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.

Bit/Field	Name	Type	Reset	Description
10	CBEIM	RW	0	GPTM Timer B Capture Mode Event Interrupt Mask The `CBEIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
9	CBMIM	RW	0	GPTM Timer B Capture Mode Match Interrupt Mask The `CBMIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
8	TBTOIM	RW	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	0	Software should not rely on the value of 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	RW	0	GPTM Timer A Match Interrupt Mask The `TAMIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
3	RTCIM	RW	0	GPTM RTC Interrupt Mask The `RTCIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
2	CAEIM	RW	0	GPTM Timer A Capture Mode Event Interrupt Mask The `CAEIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.

Bit/Field	Name	Type	Reset	Description
1	CAMIM	RW	0	GPTM Timer A Capture Mode Match Interrupt Mask The `CAMIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
0	TATOIM	RW	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 7: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C

This register shows the state of the GPTM's internal interrupt signal. These bits are set whether or not the interrupt is masked in the **GPTMIMR** register. Each bit can be cleared by writing a 1 to its corresponding bit in **GPTMICR**.

Note: The state of the **GPTMRIS** register is not affected by disabling and then re-enabling the timer using the `TnEN` bits in the **GPTM Control (GPTMCTL)** register. If an application requires that all or certain status bits should not carry over after re-enabling the timer, then the appropriate bits in the **GPTMRIS** register should be cleared using the **GPTMICR** register prior to re-enabling the timer. If this is not done, any status bits set in the **GPTMRIS** register and unmasked in the **GPTMIMR** register generate an interrupt once the timer is re-enabled.

GPTM Raw Interrupt Status (GPTMRIS)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x01C
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															WUERIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TBMRIS	CBERIS	CBMRIS	TBTORIS	reserved			TAMRIS	RTCRIS	CAERIS	CAMRIS	TATORIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	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.
16	WUERIS	RW	0	32/64-Bit Wide GPTM Write Update Error Raw Interrupt Status Value Description 0 No error. 1 Either a Timer A register or a Timer B register was written twice in a row or a Timer A register was written before the corresponding Timer B register was written.
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
11	TBMRIS	RO	0	GPTM Timer B Match Raw Interrupt Value Description 0 The match value has not been reached. 1 The TBMIE bit is set in the **GPTMTBMR** register, and the match values in the **GPTMTBMATCHR** and (optionally) **GPTMTBPMR** registers have been reached when configured in one-shot or periodic mode. This bit is cleared by writing a 1 to the TBMCINT bit in the **GPTMICR** register.
10	CBERIS	RO	0	GPTM Timer B Capture Mode Event Raw Interrupt Value Description 0 The capture mode event for Timer B has not occurred. 1 A capture mode event has occurred for Timer B. This interrupt asserts when the subtimer is configured in Input Edge-Time mode or when configured in PWM mode with the PWM interrupt enabled by setting the TBPWMIE bit in the **GPTMTBMR**. This bit is cleared by writing a 1 to the CBECINT bit in the **GPTMICR** register.
9	CBMRIS	RO	0	GPTM Timer B Capture Mode Match Raw Interrupt Value Description 0 The capture mode match for Timer B has not occurred. 1 The capture mode match has occurred for Timer B. This interrupt asserts when the values in the **GPTMTBR** and **GPTMTBPR** match the values in the **GPTMTBMATCHR** and **GPTMTBPMR** when configured in Input Edge-Time mode. This bit is cleared by writing a 1 to the CBMCINT bit in the **GPTMICR** register.
8	TBTORIS	RO	0	GPTM Timer B Time-Out Raw Interrupt Value Description 0 Timer B has not timed out. 1 Timer B has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches it's count limit (0 or the value loaded into **GPTMTBILR**, depending on the count direction). This bit is cleared by writing a 1 to the TBTOCINT bit in the **GPTMICR** register.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
4	TAMRIS	RO	0	GPTM Timer A Match Raw Interrupt Value Description 0 The match value has not been reached. 1 The `TAMIE` bit is set in the **GPTMTAMR** register, and the match value in the **GPTMTAMATCHR** and (optionally) **GPTMTAPMR** registers have been reached when configured in one-shot or periodic mode. This bit is cleared by writing a 1 to the `TAMCINT` bit in the **GPTMICR** register.
3	RTCRIS	RO	0	GPTM RTC Raw Interrupt Value Description 0 The RTC event has not occurred. 1 The RTC event has occurred. This bit is cleared by writing a 1 to the `RTCCINT` bit in the **GPTMICR** register.
2	CAERIS	RO	0	GPTM Timer A Capture Mode Event Raw Interrupt Value Description 0 The capture mode event for Timer A has not occurred. 1 A capture mode event has occurred for Timer A. This interrupt asserts when the subtimer is configured in Input Edge-Time mode or when configured in PWM mode with the PWM interrupt enabled by setting the `TAPWMIE` bit in the **GPTMTAMR**. This bit is cleared by writing a 1 to the `CAECINT` bit in the **GPTMICR** register.
1	CAMRIS	RO	0	GPTM Timer A Capture Mode Match Raw Interrupt Value Description 0 The capture mode match for Timer A has not occurred. 1 A capture mode match has occurred for Timer A. This interrupt asserts when the values in the **GPTMTAR** and **GPTMTAPR** match the values in the **GPTMTAMATCHR** and **GPTMTAPMR** when configured in Input Edge-Time mode. This bit is cleared by writing a 1 to the `CAMCINT` bit in the **GPTMICR** register.
0	TATORIS	RO	0	GPTM Timer A Time-Out Raw Interrupt Value Description 0 Timer A has not timed out. 1 Timer A has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches it's count limit (0 or the value loaded into **GPTMTAILR**, depending on the count direction). This bit is cleared by writing a 1 to the `TATOCINT` bit in the **GPTMICR** register.

Register 8: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020

This register show the state of the GPTM's controller-level interrupt. If an interrupt is unmasked in **GPTMIMR**, and there is an event that causes the interrupt to be asserted, the corresponding bit is set in this register. All bits are cleared by writing a 1 to the corresponding bit in **GPTMICR**.

GPTM Masked Interrupt Status (GPTMMIS)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x020
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															WUEMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	RO	RO	RO	RO	RO	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	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.
16	WUEMIS	RO	0	32/64-Bit Wide GPTM Write Update Error Masked Interrupt Status Value Description 0 An unmasked Write Update Error has not occurred. 1 An unmasked Write Update Error has occurred.
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMMIS	RO	0	GPTM Timer B Match Masked Interrupt Value Description 0 A Timer B Mode Match interrupt has not occurred or is masked. 1 An unmasked Timer B Mode Match interrupt has occurred. This bit is cleared by writing a 1 to the `TBMCINT` bit in the **GPTMICR** register.

Bit/Field	Name	Type	Reset	Description
10	CBEMIS	RO	0	GPTM Timer B Capture Mode Event Masked Interrupt Value Description 0 A Capture B event interrupt has not occurred or is masked. 1 An unmasked Capture B event interrupt has occurred. This bit is cleared by writing a 1 to the `CBECINT` bit in the **GPTMICR** register.
9	CBMMIS	RO	0	GPTM Timer B Capture Mode Match Masked Interrupt Value Description 0 A Capture B Mode Match interrupt has not occurred or is masked. 1 An unmasked Capture B Match interrupt has occurred. This bit is cleared by writing a 1 to the `CBMCINT` bit in the **GPTMICR** register.
8	TBTOMIS	RO	0	GPTM Timer B Time-Out Masked Interrupt Value Description 0 A Timer B Time-Out interrupt has not occurred or is masked. 1 An unmasked Timer B Time-Out interrupt has occurred. This bit is cleared by writing a 1 to the `TBTOCINT` bit in the **GPTMICR** register.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMMIS	RO	0	GPTM Timer A Match Masked Interrupt Value Description 0 A Timer A Mode Match interrupt has not occurred or is masked. 1 An unmasked Timer A Mode Match interrupt has occurred. This bit is cleared by writing a 1 to the `TAMCINT` bit in the **GPTMICR** register.
3	RTCMIS	RO	0	GPTM RTC Masked Interrupt Value Description 0 An RTC event interrupt has not occurred or is masked. 1 An unmasked RTC event interrupt has occurred. This bit is cleared by writing a 1 to the `RTCCINT` bit in the **GPTMICR** register.

Bit/Field	Name	Type	Reset	Description
2	CAEMIS	RO	0	GPTM Timer A Capture Mode Event Masked Interrupt Value Description 0 A Capture A event interrupt has not occurred or is masked. 1 An unmasked Capture A event interrupt has occurred. This bit is cleared by writing a 1 to the `CAECINT` bit in the **GPTMICR** register.
1	CAMMIS	RO	0	GPTM Timer A Capture Mode Match Masked Interrupt Value Description 0 A Capture A Mode Match interrupt has not occurred or is masked. 1 An unmasked Capture A Match interrupt has occurred. This bit is cleared by writing a 1 to the `CAMCINT` bit in the **GPTMICR** register.
0	TATOMIS	RO	0	GPTM Timer A Time-Out Masked Interrupt Value Description 0 A Timer A Time-Out interrupt has not occurred or is masked. 1 An unmasked Timer A Time-Out interrupt has occurred. This bit is cleared by writing a 1 to the `TATOCINT` bit in the **GPTMICR** register.

Register 9: GPTM Interrupt Clear (GPTMICR), offset 0x024

This register is used to clear the status bits in the **GPTMRIS** and **GPTMMIS** registers. Writing a 1 to a bit clears the corresponding bit in the **GPTMRIS** and **GPTMMIS** registers.

GPTM Interrupt Clear (GPTMICR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x024
Type W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															WUECINT
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				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
31:17	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.
16	WUECINT	RW	0	32/64-Bit Wide GPTM Write Update Error Interrupt Clear Writing a 1 to this bit clears the `WUERIS` bit in the **GPTMRIS** register and the `WUEMIS` bit in the **GPTMMIS** register.
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMCINT	W1C	0	GPTM Timer B 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 Timer B Capture Mode 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 Timer B Capture Mode 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.

Bit/Field	Name	Type	Reset	Description
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMCINT	W1C	0	GPTM Timer A 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.
2	CAECINT	W1C	0	GPTM Timer A Capture Mode 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 Timer A Capture Mode 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 10: GPTM Timer A Interval Load (GPTMTAILR), offset 0x028

When the timer is counting down, this register is used to load the starting count value into the timer. When the timer is counting up, this register sets the upper bound for the timeout event.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, **GPTMTAILR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Interval Load (GPTMTBILR)** register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBILR**.

When a 32/64-bit Wide GPTM is configured to one of the 64-bit modes, **GPTMTAILR** contains bits 31:0 of the 64-bit count and the **GPTM Timer B Interval Load (GPTMTBILR)** register contains bits 63:32.

GPTM Timer A Interval Load (GPTMTAILR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x028
Type RW, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAILR															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	TAILR	RW	0xFFFF.FFFF	GPTM Timer A Interval Load Register Writing this field loads the counter for Timer A. A read returns the current value of **GPTMTAILR**.

Register 11: GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C

When the timer is counting down, this register is used to load the starting count value into the timer. When the timer is counting up, this register sets the upper bound for the timeout event.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAILR** register. Reads from this register return the current value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the load value. Bits 31:16 are reserved in both cases.

When a 32/64-bit Wide GPTM is configured to one of the 64-bit modes, **GPTMTAILR** contains bits 31:0 of the 64-bit count and the **GPTMTBILR** register contains bits 63:32.

GPTM Timer B Interval Load (GPTMTBILR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x02C
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TBILR															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	TBILR	RW	0x0000.FFFF (for 16/32-bit) 0xFFFF.FFFF (for 32/64-bit)	GPTM Timer B Interval Load Register Writing this field loads the counter for Timer B. A read returns the current value of **GPTMTBILR**. When a 16/32-bit GPTM is in 32-bit mode, writes are ignored, and reads return the current value of **GPTMTBILR**.

Register 12: GPTM Timer A Match (GPTMTAMATCHR), offset 0x030

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode.

In Edge-Count mode, this register along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value. Note that in edge-count mode, when executing an up-count, the value of **GPTMTnPR** and **GPTMTnILR** must be greater than the value of **GPTMTnPMR** and **GPTMTnMATCHR**.

In PWM mode, this value along with **GPTMTAILR**, determines the duty cycle of the output PWM signal.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, **GPTMTAMATCHR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Match (GPTMTBMATCHR)** register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBMATCHR**.

When a 32/64-bit Wide GPTM is configured to one of the 64-bit modes, **GPTMTAMATCHR** contains bits 31:0 of the 64-bit match value and the **GPTM Timer B Match (GPTMTBMATCHR)** register contains bits 63:32.

GPTM Timer A Match (GPTMTAMATCHR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x030
Type RW, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAMR															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	TAMR	RW	0xFFFF.FFFF	GPTM Timer A Match Register This value is compared to the **GPTMTAR** register to determine match events.

Register 13: GPTM Timer B Match (GPTMTBMATCHR), offset 0x034

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode.

In Edge-Count mode, this register along with **GPTMTBILR** determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTBILR** minus this value. Note that in edge-count mode, when executing an up-count, the value of **GPTMTnPR** and **GPTMTnILR** must be greater than the value of **GPTMTnPMR** and **GPTMTnMATCHR**.

In PWM mode, this value along with **GPTMTBILR**, determines the duty cycle of the output PWM signal.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAMATCHR** register. Reads from this register return the current match value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the match value. Bits 31:16 are reserved in both cases.

When a 32/64-bit Wide GPTM is configured to one of the 64-bit modes, **GPTMTAMATCHR** contains bits 31:0 of the 64-bit match value and the **GPTMTBMATCHR** register contains bits 63:32.

GPTM Timer B Match (GPTMTBMATCHR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x034
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TBMR															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	TBMR	RW	0x0000.FFFF (for 16/32-bit) 0xFFFF.FFFF (for 32/64-bit)	GPTM Timer B Match Register This value is compared to the **GPTMTBR** register to determine match events.

Register 14: GPTM Timer A Prescale (GPTMTAPR), offset 0x038

This register allows software to extend the range of the timers when they are used individually. When in one-shot or periodic down count modes, this register acts as a true prescaler for the timer counter. When acting as a true prescaler, the prescaler counts down to 0 before the value in the **GPTMTAR** and **GPTMTAV** registers are incremented. In all other individual/split modes, this register is a linear extension of the upper range of the timer counter, holding bits 23:16 in the 16-bit modes of the 16/32-bit GPTM and bits 47:32 in the 32-bit modes of the 32/64-bit Wide GPTM.

GPTM Timer A Prescale (GPTMTAPR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x038
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TAPSRH								TAPSR							
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of 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	TAPSRH	RW	0x00	GPTM Timer A Prescale High Byte The register loads this value on a write. A read returns the current value of the register. For the 16/32-bit GPTM, this field is reserved. For the 32/64-bit Wide GPTM, this field contains the upper 8-bits of the 16-bit prescaler. Refer to Table 10-5 on page 642 for more details and an example.
7:0	TAPSR	RW	0x00	GPTM Timer A Prescale The register loads this value on a write. A read returns the current value of the register. For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler. For the 32/64-bit Wide GPTM, this field contains the lower 8-bits of the 16-bit prescaler. Refer to Table 10-5 on page 642 for more details and an example.

Register 15: GPTM Timer B Prescale (GPTMTBPR), offset 0x03C

This register allows software to extend the range of the timers when they are used individually. When in one-shot or periodic down count modes, this register acts as a true prescaler for the timer counter. When acting as a true prescaler, the prescaler counts down to 0 before the value in the **GPTMTBR** and **GPTMTBV** registers are incremented. In all other individual/split modes, this register is a linear extension of the upper range of the timer counter, holding bits 23:16 in the 16-bit modes of the 16/32-bit GPTM and bits 47:32 in the 32-bit modes of the 32/64-bit Wide GPTM.

GPTM Timer B Prescale (GPTMTBPR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x03C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TBPSRH								TBPSR							
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of 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	TBPSRH	RW	0x00	GPTM Timer B Prescale High Byte The register loads this value on a write. A read returns the current value of the register. For the 16/32-bit GPTM, this field is reserved. For the 32/64-bit Wide GPTM, this field contains the upper 8-bits of the 16-bit prescaler. Refer to Table 10-5 on page 642 for more details and an example.
7:0	TBPSR	RW	0x00	GPTM Timer B Prescale The register loads this value on a write. A read returns the current value of this register. For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler. For the 32/64-bit Wide GPTM, this field contains the lower 8-bits of the 16-bit prescaler. Refer to Table 10-5 on page 642 for more details and an example.

Register 16: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040

This register allows software to extend the range of the **GPTMTAMATCHR** when the timers are used individually. This register holds bits 23:16 in the 16-bit modes of the 16/32-bit GPTM and bits 47:32 in the 32-bit modes of the 32/64-bit Wide GPTM.

GPTM TimerA Prescale Match (GPTMTAPMR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x040
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TAPSMRH								TAPSMR							
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	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	TAPSMRH	RW	0x00	GPTM Timer A Prescale Match High Byte This value is used alongside **GPTMTAMATCHR** to detect timer match events while using a prescaler. For the 16/32-bit GPTM, this field is reserved. For the 32/64-bit Wide GPTM, this field contains the upper 8-bits of the 16-bit prescale match value.
7:0	TAPSMR	RW	0x00	GPTM TimerA Prescale Match This value is used alongside **GPTMTAMATCHR** to detect timer match events while using a prescaler. For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler match value. For the 32/64-bit Wide GPTM, this field contains the lower 8-bits of the 16-bit prescaler match value.

Register 17: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044

This register allows software to extend the range of the **GPTMTBMATCHR** when the timers are used individually. This register holds bits 23:16 in the 16-bit modes of the 16/32-bit GPTM and bits 47:32 in the 32-bit modes of the 32/64-bit Wide GPTM.

GPTM TimerB Prescale Match (GPTMTBPMR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x044
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TBPSMRH								TBPSMR							
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	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	TBPSMRH	RW	0x00	GPTM Timer B Prescale Match High Byte This value is used alongside **GPTMTBMATCHR** to detect timer match events while using a prescaler. For the 16/32-bit GPTM, this field is reserved. For the 32/64-bit Wide GPTM, this field contains the upper 8-bits of the 16-bit prescale match value.
7:0	TBPSMR	RW	0x00	GPTM TimerB Prescale Match This value is used alongside **GPTMTBMATCHR** to detect timer match events while using a prescaler. For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler match value. For the 32/64-bit Wide GPTM, this field contains the lower 8-bits of the 16-bit prescaler match value.

Register 18: GPTM Timer A (GPTMTAR), offset 0x048

This register shows the current value of the Timer A counter in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, **GPTMTAR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B (GPTMTBR)** register). In the16-bit Input Edge Count, Input Edge Time, and PWM modes, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler, which is the upper 8 bits of the count. Bits 31:24 always read as 0. To read the value of the prescaler in 16-bit One-Shot and Periodic modes, read bits [23:16] in the **GPTMTAV** register. To read the value of the prescalar in periodic snapshot mode, read the **Timer A Prescale Snapshot (GPTMTAPS)** register.

When a 32/64-bit Wide GPTM is configured to one of the 64-bit modes, **GPTMTAR** contains bits 31:0 of the 64-bit timer value and the **GPTM Timer B (GPTMTBR)** register contains bits 63:32. In a 32-bit mode, the value of the prescaler is stored in the **GPTM Timer A Prescale Snapshot (GPTMTAPS)** register.

GPTM Timer A (GPTMTAR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x048
Type RO, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAR															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TAR															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:0	TAR	RO	0xFFFF.FFFF	GPTM Timer A Register A read returns the current value of the **GPTM Timer A Count Register**, in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

Register 19: GPTM Timer B (GPTMTBR), offset 0x04C

This register shows the current value of the Timer B counter in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAR** register. Reads from this register return the current value of Timer B. In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler in Input Edge Count, Input Edge Time, and PWM modes, which is the upper 8 bits of the count. Bits 31:24 always read as 0. To read the value of the prescaler in 16-bit One-Shot and Periodic modes, read bits [23:16] in the **GPTMTBV** register. To read the value of the prescalar in periodic snapshot mode, read the **Timer B Prescale Snapshot (GPTMTBPS)** register.

When a 32/64-bit Wide GPTM is configured to one of the 64-bit modes, **GPTMTAR** contains bits 31:0 of the 64-bit timer value and the **GPTM Timer B (GPTMTBR)** register contains bits 63:32. In a 32-bit mode, the value of the prescaler is stored in the **GPTM Timer B Prescale Snapshot (GPTMTBPS)** register.

GPTM Timer B (GPTMTBR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x04C
Type RO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TBR															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TBR															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	1	0	0	1	1	0	0	1	0	0	1	0	1	1

Bit/Field	Name	Type	Reset	Description
31:0	TBR	RO	0x0000.FFFF (for 16/32-bit) 0xFFFF.FFFF (for 32/64-bit)	GPTM Timer B Register A read returns the current value of the **GPTM Timer B Count Register**, in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

Register 20: GPTM Timer A Value (GPTMTAV), offset 0x050

When read, this register shows the current, free-running value of Timer A in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry when using the snapshot feature with the periodic operating mode. When written, the value written into this register is loaded into the **GPTMTAR** register on the next clock cycle.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, **GPTMTAV** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Value (GPTMTBV)** register). In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count in Input Edge Count, Input Edge Time, PWM and one-shot or periodic up count modes. In one-shot or periodic down count modes, the prescaler stored in 23:16 is a true prescaler, meaning bits 23:16 count down before decrementing the value in bits 15:0. The prescaler in bits 31:24 always reads as 0.

When a 32/64-bit Wide GPTM is configured to one of the 64-bit modes, **GPTMTAV** contains bits 31:0 of the 64-bit timer value and the **GPTM Timer B Value (GPTMTBV)** register contains bits 63:32. In a 32-bit mode, the current, free-running value of the prescaler is stored in the **GPTM Timer A Prescale Value (GPTMTAPV)** register.mint

GPTM Timer A Value (GPTMTAV)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x050
Type RW, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAV															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	TAV	RW	0xFFFF.FFFF	GPTM Timer A Value A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the **GPTMTAR** register on the next clock cycle. **Note:** In 16-bit mode, only the lower 16-bits of the **GPTMTAV** register can be written with a new value. Writes to the prescaler bits have no effect.

Register 21: GPTM Timer B Value (GPTMTBV), offset 0x054

When read, this register shows the current, free-running value of Timer B in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry. When written, the value written into this register is loaded into the **GPTMTBR** register on the next clock cycle.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAV** register. Reads from this register return the current free-running value of Timer B. In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count in Input Edge Count, Input Edge Time, PWM and one-shot or periodic up count modes. In one-shot or periodic down count modes, the prescaler stored in 23:16 is a true prescaler, meaning bits 23:16 count down before decrementing the value in bits 15:0. The prescaler in bits 31:24 always reads as 0.

When a 32/64-bit Wide GPTM is configured to one of the 64-bit modes, **GPTMTBV** contains bits 63:32 of the 64-bit timer value and the **GPTM Timer A Value (GPTMTAV)** register contains bits 31:0. In a 32-bit mode, the current, free-running value of the prescaler is stored in the **GPTM Timer B Prescale Value (GPTMTBPV)** register.

GPTM Timer B Value (GPTMTBV)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x054
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TBV															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	TBV	RW	0x0000.FFFF (for 16/32-bit) 0xFFFF.FFFF (for 32/64-bit)	GPTM Timer B Value A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the **GPTMTAR** register on the next clock cycle. **Note:** In 16-bit mode, only the lower 16-bits of the **GPTMTBV** register can be written with a new value. Writes to the prescaler bits have no effect.

Register 22: GPTM RTC Predivide (GPTMRTCPD), offset 0x058

This register provides the current RTC predivider value when the timer is operating in RTC mode. Software must perform an atomic access with consecutive reads of the **GPTMTAR**, **GPTMTBR**, and **GPTMRTCPD** registers, see Figure 10-2 on page 644 for more information.

GPTM RTC Predivide (GPTMRTCPD)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x058
Type RO, 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
	RTCPD															
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	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	RTCPD	RO	0x0000.7FFF	RTC Predivide Counter Value The current RTC predivider value when the timer is operating in RTC mode. This field has no meaning in other timer modes.

Register 23: GPTM Timer A Prescale Snapshot (GPTMTAPS), offset 0x05C

For the 32/64-bit Wide GPTM, this register shows the current value of the Timer A prescaler in the 32-bit modes. For 16-/32-bit wide GPTM, this register shows the current value of the Timer A prescaler for periodic snapshot mode.

GPTM Timer A Prescale Snapshot (GPTMTAPS)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x05C
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
	PSS															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	PSS	RO	0x0000	GPTM Timer A Prescaler Snapshot A read returns the current value of the **GPTM Timer A Prescaler.**

Register 24: GPTM Timer B Prescale Snapshot (GPTMTBPS), offset 0x060

For the 32/64-bit Wide GPTM, this register shows the current value of the Timer B prescaler in the 32-bit modes. For 16-/32-bit wide GPTM, this register shows the current value of the Timer B prescaler for periodic snapshot mode.

GPTM Timer B Prescale Snapshot (GPTMTBPS)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
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
	PSS															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	PSS	RO	0x0000	GPTM Timer A Prescaler Value A read returns the current value of the **GPTM Timer A Prescaler.**

Register 25: GPTM Timer A Prescale Value (GPTMTAPV), offset 0x064

For the 32/64-bit Wide GPTM, this register shows the current free-running value of the Timer A prescaler in the 32-bit modes. Software can use this value in conjunction with the **GPTMTAV** register to determine the time elapsed between an interrupt and the ISR entry. This register is ununsed in 16/32-bit GPTM mode.

GPTM Timer A Prescale Value (GPTMTAPV)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x064
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
	PSV															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	PSV	RO	0x0000	GPTM Timer A Prescaler Value A read returns the current, free-running value of the Timer A prescaler.

Register 26: GPTM Timer B Prescale Value (GPTMTBPV), offset 0x068

For the 32/64-bit Wide GPTM, this register shows the current free-running value of the Timer B prescaler in the 32-bit modes. Software can use this value in conjunction with the **GPTMTBV** register to determine the time elapsed between an interrupt and the ISR entry. This register is ununsed in 16/32-bit GPTM mode.

GPTM Timer B Prescale Value (GPTMTBPV)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0x068
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
	PSV															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	PSV	RO	0x0000	GPTM Timer B Prescaler Value A read returns the current, free-running value of the Timer A prescaler.

Register 27: GPTM Peripheral Properties (GPTMPP), offset 0xFC0

The **GPTMPP** register provides information regarding the properties of the General-Purpose Timer module.

GPTM Peripheral Properties (GPTMPP)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
32/64-bit Wide Timer 0 base: 0x4003.6000
32/64-bit Wide Timer 1 base: 0x4003.7000
32/64-bit Wide Timer 2 base: 0x4004.C000
32/64-bit Wide Timer 3 base: 0x4004.D000
32/64-bit Wide Timer 4 base: 0x4004.E000
32/64-bit Wide Timer 5 base: 0x4004.F000
Offset 0xFC0
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												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	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	SIZE	RO	0x0	Count Size

Value	Description
0	Timer A and Timer B counters are 16 bits each with an 8-bit prescale counter.
1	Timer A and Timer B counters are 32 bits each with a 16-bit prescale counter.

11 Watchdog Timers

A watchdog timer can generate a non-maskable interrupt (NMI), a regular interrupt or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or due to the failure of an external device to respond in the expected way. The TM4C1232C3PM microcontroller has two Watchdog Timer Modules, one module is clocked by the system clock (Watchdog Timer 0) and the other (Watchdog Timer 1) is clocked by the PIOSC 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.

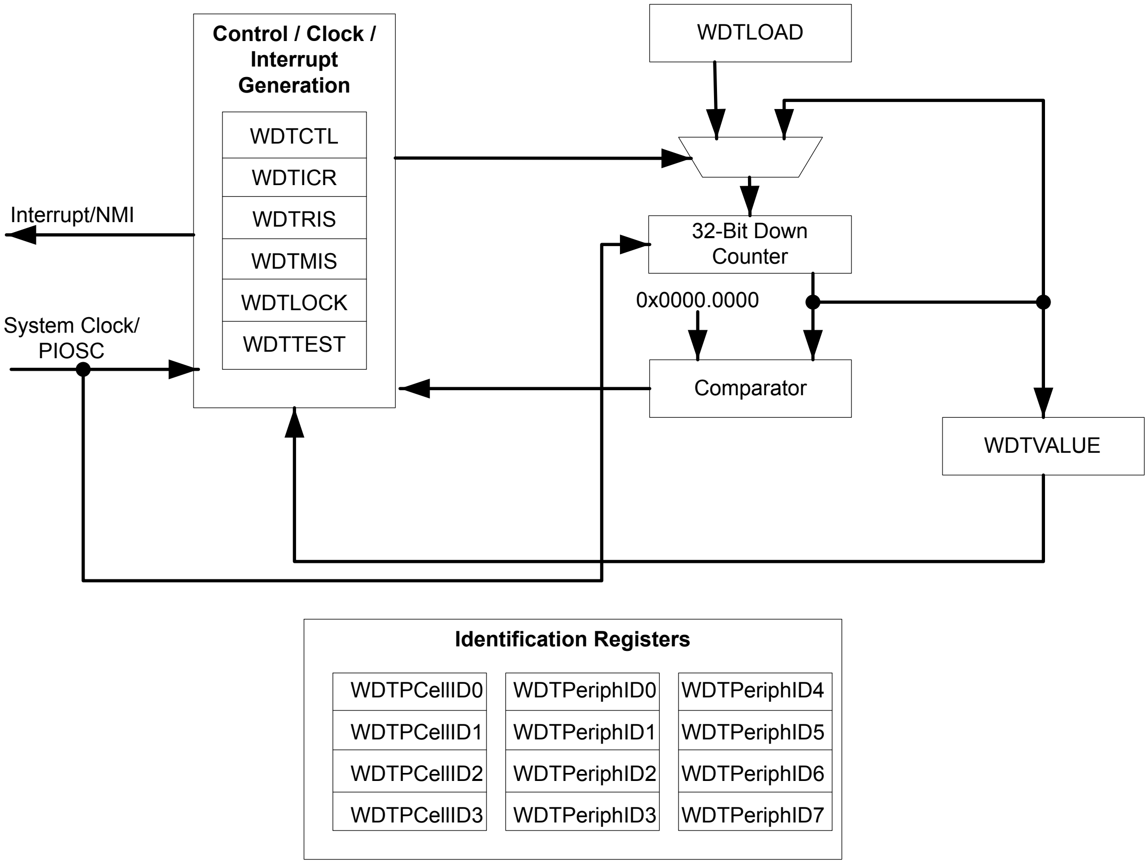
The TM4C1232C3PM controller has two Watchdog Timer modules with the following features:

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

The Watchdog Timer can be configured to generate an interrupt to the controller on its first time-out, and to generate a reset signal on its second time-out. Once the Watchdog Timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

11.1 Block Diagram

Figure 11-1. WDT Module Block Diagram



11.2 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. The watchdog interrupt can be programmed to be a non-maskable interrupt (NMI) using the `INTTYPE` bit in the **WDTCTL** register. After the first time-out event, the 32-bit counter is re-loaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. Once the Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled by setting the `RESEN` bit in the **WDTCTL** register, the Watchdog timer asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the **WDTLOAD** register, and counting resumes from that value.

If **WDTLOAD** is written with a new value while the Watchdog Timer counter is counting, then the counter is loaded with the new value and continues counting.

Writing to **WDTLOAD** does not clear an active interrupt. An interrupt must be specifically cleared by writing to the **Watchdog Interrupt Clear (WDTICR)** register.

The Watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is re-enabled, the 32-bit counter is preloaded with the load register value and not its last state.

The watchdog timer is disabled by default out of reset. To achieve maximum watchdog protection of the device, the watchdog timer can be enabled at the start of the reset vector.

11.2.1 Register Access Timing

Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The WRC bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for WRC=1 prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock.

11.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the Rn bit in the **Watchdog Timer Run Mode Clock Gating Control (RCGCWD)** register, see page 316.

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, enable interrupts, 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.

To service the watchdog, periodically reload the count value into the **WDTLOAD** register to restart the count. The interrupt can be enabled using the INTEN bit in the **WDTCTL** register to allow the processor to attempt corrective action if the watchdog is not serviced often enough. The RESEN bit in **WDTCTL** can be set so that the system resets if the failure is not recoverable using the ISR.

11.4 Register Map

Table 11-1 on page 709 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 316).

Table 11-1. Watchdog Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	RW	0xFFFF.FFFF	Watchdog Load	710
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	711
0x008	WDTCTL	RW	0x0000.0000 (WDT0) 0x8000.0000 (WDT1)	Watchdog Control	712
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	714
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	715
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	716
0x418	WDTTEST	RW	0x0000.0000	Watchdog Test	717
0xC00	WDTLOCK	RW	0x0000.0000	Watchdog Lock	718
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	719
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	720
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	721
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	722
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	723
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	724
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	725
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	726
0xFF0	WDTPCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	727
0xFF4	WDTPCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	728
0xFF8	WDTPCellID2	RO	0x0000.0006	Watchdog PrimeCell Identification 2	729
0xFFC	WDTPCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	730

11.5 Register Descriptions

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

Register 1: Watchdog Load (WDTLOAD), offset 0x000

This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter restarts counting down from the new value. If the **WDTLOAD** register is loaded with 0x0000.0000, an interrupt is immediately generated.

Watchdog Load (WDTLOAD)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x000
Type RW, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WDTLOAD															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Bit/Field	Name	Type	Reset	Description
31:0	WDTLOAD	RW	0xFFFF.FFFF	Watchdog Load Value

Register 2: Watchdog Value (WDTVALUE), offset 0x004

This register contains the current count value of the timer.

Watchdog Value (WDTVALUE)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x004
Type RO, reset 0xFFFF.FFFF

	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 by setting the `INTEN` bit, all subsequent writes to the `INTEN` bit are ignored. The only mechanisms that can re-enable writes to this bit are a hardware reset or a software reset initiated by setting the appropriate bit in the **Watchdog Timer Software Reset (SRWD)** register.

Important: Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The `WRC` bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for `WRC`=1 prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock and therefore does not have a `WRC` bit.

Watchdog Control (WDTCTL)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x008
Type RW, reset 0x0000.0000 (WDT0) and 0x8000.0000 (WDT1)

	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													INTTYPE	RESEN	INTEN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	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.
30:3	reserved	RO	0x000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2	INTTYPE	RW	0	Watchdog Interrupt Type The `INTTYPE` values are defined as follows: Value Description 0 Watchdog interrupt is a standard interrupt. 1 Watchdog interrupt is a non-maskable interrupt.
1	RESEN	RW	0	Watchdog Reset Enable The `RESEN` values are defined as follows: Value Description 0 Disabled. 1 Enable the Watchdog module reset output.
0	INTEN	RW	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 or a software reset initiated by setting the appropriate bit in the **Watchdog Timer Software Reset (SRWD)** register. 1 Interrupt event enabled. Once enabled, all writes are ignored. Setting this bit enables the Watchdog Timer.

Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C

This register is the interrupt clear register. A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the **WDTLOAD** register. Write to this register when a watchdog time-out interrupt has occurred to properly service the Watchdog. Value for a read or reset is indeterminate.

Note: Locking the watchdog registers by using the **WDTLOCK** register does not affect the **WDTICR** register and allows interrupts to always be serviced. Thus, a write at any time of the **WDTICR** register clears the **WDTMIS** register and reloads the 32-bit counter from the **WDTLOAD** register. The **WDTICR** register should only be written when interrupts have triggered and need to be serviced.

Watchdog Interrupt Clear (WDTICR)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x00C
Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WDTINTCLR															
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
	WDTINTCLR															
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	WDTINTCLR	WO	-	Watchdog Interrupt Clear A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the **WDTLOAD** register. Write to this register when a watchdog time-out interrupt has occurred to properly service the Watchdog. Value for a read or reset is indeterminate.

Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010

This register is the raw interrupt status register. Watchdog interrupt events can be monitored via this register if the controller interrupt is masked.

Watchdog Raw Interrupt Status (WDTRIS)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x010
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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 0 The watchdog has not timed out. 1 A watchdog time-out event has occurred.

Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014

This register is the masked interrupt status register. The value of this register is the logical AND of the raw interrupt bit and the Watchdog interrupt enable bit.

Watchdog Masked Interrupt Status (WDTMIS)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x014
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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
0	The watchdog has not timed out or the watchdog timer interrupt is masked.
1	A watchdog time-out event has been signalled to the interrupt controller.

Register 7: Watchdog Test (WDTTEST), offset 0x418

This register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

Watchdog Test (WDTTEST)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x418
Type RW, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	STALL	RW	0	Watchdog Stall Enable Value Description 0 The watchdog timer continues counting if the microcontroller is stopped with a debugger. 1 If the microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting.
7:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: Watchdog Lock (WDTLOCK), offset 0xC00

Writing 0x1ACC.E551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers, except for the **Watchdog Test (WDTTEST)** register. The locked state will be enabled after 2 clock cycles. Reading the **WDTLOCK** register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the **WDTLOCK** register returns 0x0000.0001 (when locked; otherwise, the returned value is 0x0000.0000 (unlocked)).

Watchdog Lock (WDTLOCK)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xC00
Type RW, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:0	WDTLOCK	RW	0x0000.0000	Watchdog Lock A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates, except for the **WDTTEST** register. Avoid writes to the **WDTTEST** register when the watchdog registers are locked. 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	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]

12 Analog-to-Digital Converter (ADC)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. Two identical converter modules are included, which share 12 input channels.

The TM4C1232C3PM ADC module features 12-bit conversion resolution and supports 12 input channels, plus an internal temperature sensor. Each ADC module contains four programmable sequencers allowing the sampling of multiple analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. In addition, the conversion value can optionally be diverted to a digital comparator module. Each ADC module provides eight digital comparators. Each digital comparator evaluates the ADC conversion value against its two user-defined values to determine the operational range of the signal. The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. A phase shifter can delay the start of sampling by a specified phase angle. When using both ADC modules, it is possible to configure the converters to start the conversions coincidentally or within a relative phase from each other, see “Sample Phase Control” on page 736.

The TM4C1232C3PM microcontroller provides two ADC modules with each having the following features:

- 12 shared analog input channels
- 12-bit precision ADC
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of one million samples/second
- Optional phase shift in sample time programmable from 22.5º to 337.5º
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - GPIO
- Hardware averaging of up to 64 samples
- Eight digital comparators
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)

  - Dedicated channel for each sample sequencer
  - ADC module uses burst requests for DMA

12.1 Block Diagram

The TM4C1232C3PM microcontroller contains two identical Analog-to-Digital Converter modules. These two modules, ADC0 and ADC1, share the same 12 analog input channels. Each ADC module operates independently and can therefore execute different sample sequences, sample any of the analog input channels at any time, and generate different interrupts and triggers. Figure 12-1 on page 732 shows how the two modules are connected to analog inputs and the system bus.

Figure 12-1. Implementation of Two ADC Blocks

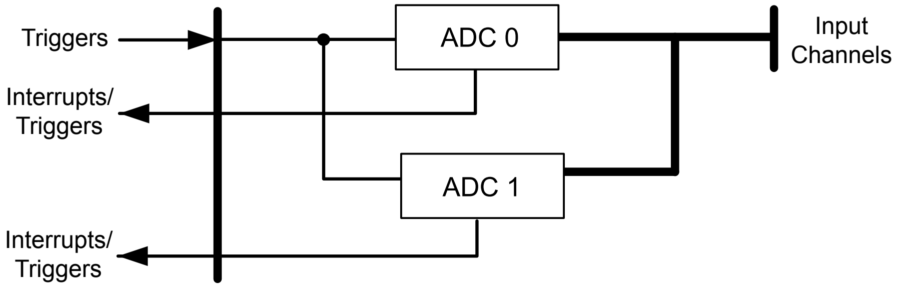
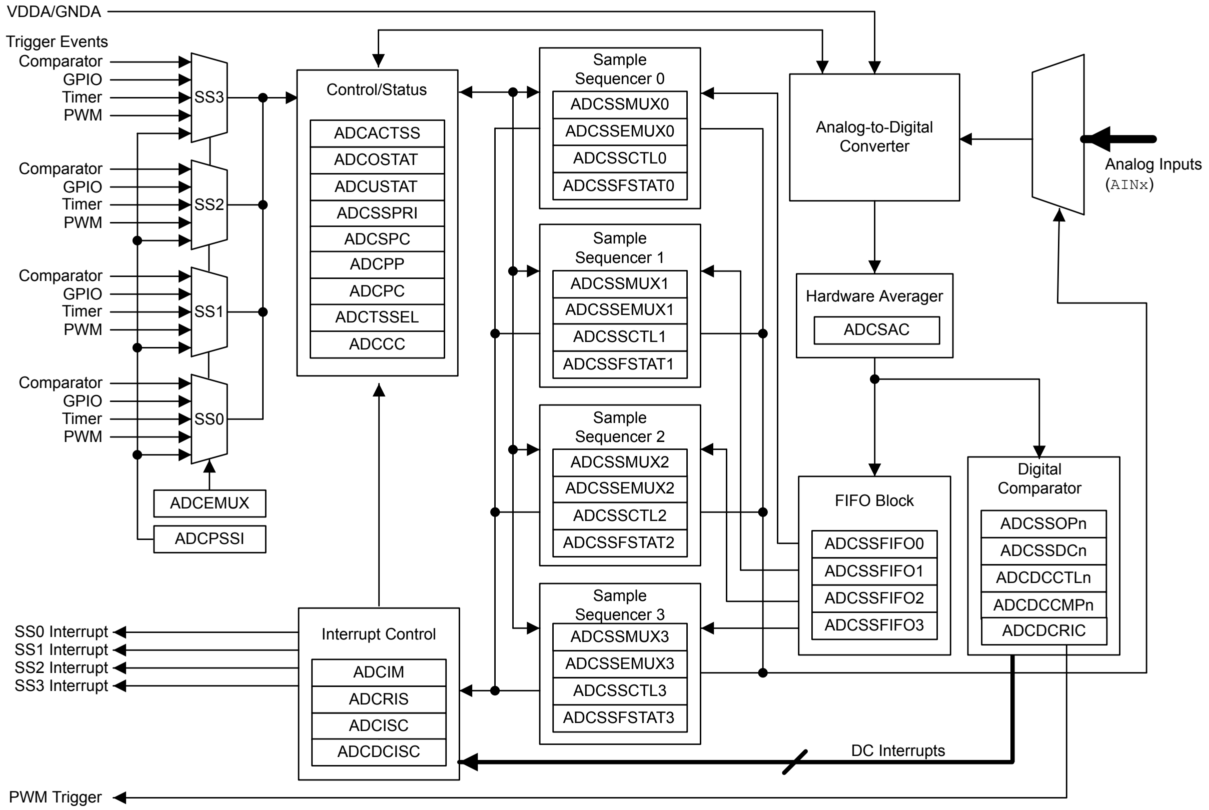


Figure 12-2 on page 733 provides details on the internal configuration of the ADC controls and data registers.

Figure 12-2. ADC Module Block Diagram



12.2 Signal Description

The following table lists the external signals of the ADC module and describes the function of each. The AINx signals are analog functions for some GPIO signals. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the ADC signals. These signals are configured by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding AMSEL bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 581.

Table 12-1. ADC Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN0	6	PE3	I	Analog	Analog-to-digital converter input 0.
AIN1	7	PE2	I	Analog	Analog-to-digital converter input 1.
AIN2	8	PE1	I	Analog	Analog-to-digital converter input 2.
AIN3	9	PE0	I	Analog	Analog-to-digital converter input 3.
AIN4	64	PD3	I	Analog	Analog-to-digital converter input 4.
AIN5	63	PD2	I	Analog	Analog-to-digital converter input 5.
AIN6	62	PD1	I	Analog	Analog-to-digital converter input 6.
AIN7	61	PD0	I	Analog	Analog-to-digital converter input 7.
AIN8	60	PE5	I	Analog	Analog-to-digital converter input 8.

Table 12-1. ADC Signals (64LQFP) ***(continued)***

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN9	59	PE4	I	Analog	Analog-to-digital converter input 9.
AIN10	58	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	57	PB5	I	Analog	Analog-to-digital converter input 11.

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

12.3 Functional Description

The TM4C1232C3PM ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approaches found on many ADC modules. Each *sample sequence* is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the processor. The programming of each sample in the sample sequence includes parameters such as the input source and mode (differential versus single-ended input), interrupt generation on sample completion, and the indicator for the last sample in the sequence. In addition, the μDMA can be used to more efficiently move data from the sample sequencers without CPU intervention.

12.3.1 Sample Sequencers

The sampling control and data capture is handled by the sample sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 12-2 on page 734 shows the maximum number of samples that each sequencer can capture and its corresponding FIFO depth. Each sample that is captured is stored in the FIFO. In this implementation, each FIFO entry is a 32-bit word, with the lower 12 bits containing the conversion result.

Table 12-2. Samples and FIFO Depth of Sequencers

Sequencer	Number of Samples	Depth of FIFO
SS3	1	1
SS2	4	4
SS1	4	4
SS0	8	8

For a given sample sequence, each sample is defined by bit fields in the **ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn)** and **ADC Sample Sequence Control (ADCSSCTLn)** registers, where "n" corresponds to the sequence number. The **ADCSSMUXn** fields select the input pin, while the **ADCSSCTLn** fields contain the sample control bits corresponding to parameters such as temperature sensor selection, interrupt enable, end of sequence, and differential input mode. Sample sequencers are enabled by setting the respective `ASENn` bit in the **ADC Active Sample Sequencer (ADCACTSS)** register and should be configured before being enabled. Sampling is then initiated by setting the `SSn` bit in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. In addition, sample sequences may be initiated on multiple ADC modules simultaneously using the `GSYNC` and `SYNCWAIT` bits in the **ADCPSSI** register during the configuration of each ADC module. For more information on using these bits, refer to page 775.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence are allowed. In the **ADCSSCTLn** register, the `IEn` bits can be set for any combination of samples, allowing interrupts to be generated after every sample in the sequence if necessary. Also, the `END` bit can be set at any point within a sample sequence. For example, if Sequencer 0 is used, the `END` bit can be set in the nibble associated with the fifth sample, allowing Sequencer 0 to complete execution of the sample sequence after the fifth sample.

After a sample sequence completes execution, the result data can be retrieved from the **ADC Sample Sequence Result FIFO (ADCSSFIFOn)** registers. The FIFOs are simple circular buffers that read a single address to "pop" result data. For software debug purposes, the positions of the FIFO head and tail pointers are visible in the **ADC Sample Sequence FIFO Status (ADCSSFSTATn)** registers along with FULL and EMPTY status flags. If a write is attempted when the FIFO is full, the write does not occur and an overflow condition is indicated. Overflow and underflow conditions are monitored using the **ADCOSTAT** and **ADCUSTAT** registers.

12.3.2 Module Control

Outside of the sample sequencers, the remainder of the control logic is responsible for tasks such as:

- Interrupt generation
- DMA operation
- Sequence prioritization
- Trigger configuration
- Comparator configuration
- Sample phase control
- Module clocking

Most of the ADC control logic runs at the ADC clock rate of 16 MHz. The internal ADC divider is configured for 16-MHz operation automatically by hardware when the system XTAL is selected with the PLL.

12.3.2.1 Interrupts

The register configurations of the sample sequencers and digital comparators dictate which events generate raw interrupts, but do not have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signals are controlled by the state of the MASK bits in the **ADC Interrupt Mask (ADCIM)** register. Interrupt status can be viewed at two locations: the **ADC Raw Interrupt Status (ADCRIS)** register, which shows the raw status of the various interrupt signals; and the **ADC Interrupt Status and Clear (ADCISC)** register, which shows active interrupts that are enabled by the **ADCIM** register. Sequencer interrupts are cleared by writing a 1 to the corresponding IN bit in **ADCISC**. Digital comparator interrupts are cleared by writing a 1 to the **ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)** register.

12.3.2.2 DMA Operation

DMA may be used to increase efficiency by allowing each sample sequencer to operate independently and transfer data without processor intervention or reconfiguration. The ADC module provides a request signal from each sample sequencer to the associated dedicated channel of the μDMA controller. The ADC does not support single transfer requests. A burst transfer request is asserted when the interrupt bit for the sample sequence is set (IE bit in the **ADCSSCTLn** register is set).

The arbitration size of the μDMA transfer must be a power of 2, and the associated IE bits in the **ADCSSCTLn** register must be set. For example, if the μDMA channel of SS0 has an arbitration size of four, the IE3 bit (4th sample) and the IE7 bit (8th sample) must be set. Thus the µDMA request occurs every time 4 samples have been acquired. No other special steps are needed to enable the ADC module for µDMA operation.

Refer to the “Micro Direct Memory Access (µDMA)” on page 517 for more details about programming the µDMA controller.

12.3.2.3 Prioritization

When sampling events (triggers) happen concurrently, they are prioritized for processing by the values in the **ADC Sample Sequencer Priority (ADCSSPRI)** register. Valid priority values are in the range of 0-3, with 0 being the highest priority and 3 being the lowest. Multiple active sample sequencer units with the same priority do not provide consistent results, so software must ensure that all active sample sequencer units have a unique priority value.

12.3.2.4 Sampling Events

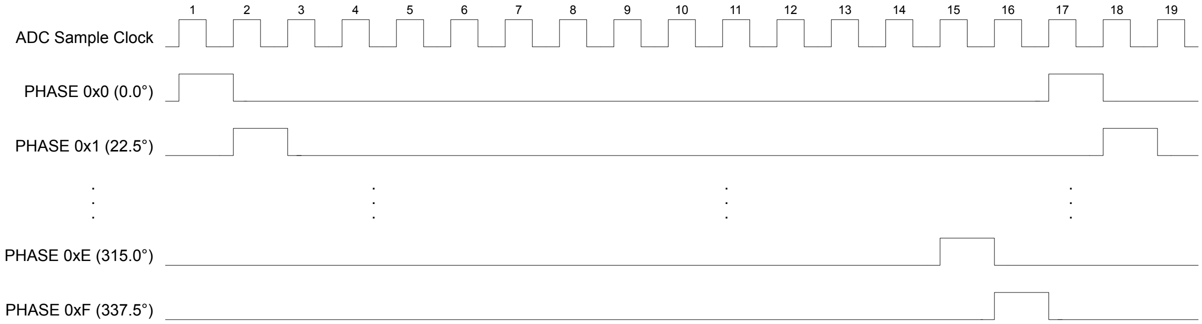
Sample triggering for each sample sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. Trigger sources include processor (default), analog comparators, an external signal on a GPIO specified by the **GPIO ADC Control (GPIOADCCTL)** register, a GP Timer, and continuous sampling. The processor triggers sampling by setting the SSx bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

Care must be taken when using the continuous sampling trigger. If a sequencer's priority is too high, it is possible to starve other lower priority sequencers. Generally, a sample sequencer using continuous sampling should be set to the lowest priority. Continuous sampling can be used with a digital comparator to cause an interrupt when a particular voltage is seen on an input.

12.3.2.5 Sample Phase Control

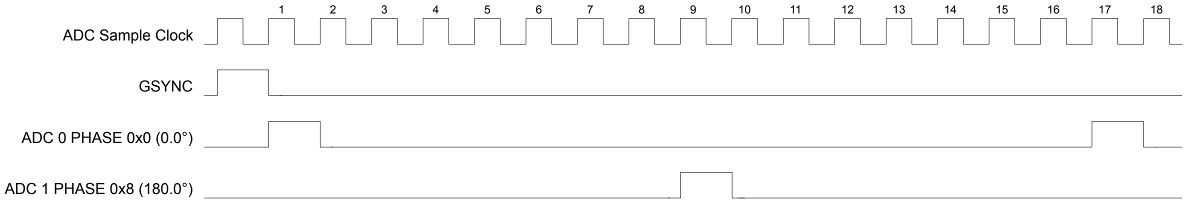
The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. If the converters are running at the same sample rate, they may be configured to start the conversions coincidentally or with one of 15 different discrete phases relative to each other. The sample time can be delayed from the standard sampling time in 22.5° increments up to 337.5º using the **ADC Sample Phase Control (ADCSPC)** register. Figure 12-3 on page 736 shows an example of various phase relationships at a 1 Msps rate.

Figure 12-3. ADC Sample Phases



This feature can be used to double the sampling rate of an input. Both ADC module 0 and ADC module 1 can be programmed to sample the same input. ADC module 0 could sample at the standard position (the PHASE field in the **ADCSPC** register is 0x0). ADC module 1 can be configured to sample at 180 (PHASE = 0x8). The two modules can be be synchronized using the GSYNC and SYNCWAIT bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. Software could then combine the results from the two modules to create a sample rate of one million samples/second at 16 MHz as shown in Figure 12-4 on page 737.

Figure 12-4. Doubling the ADC Sample Rate



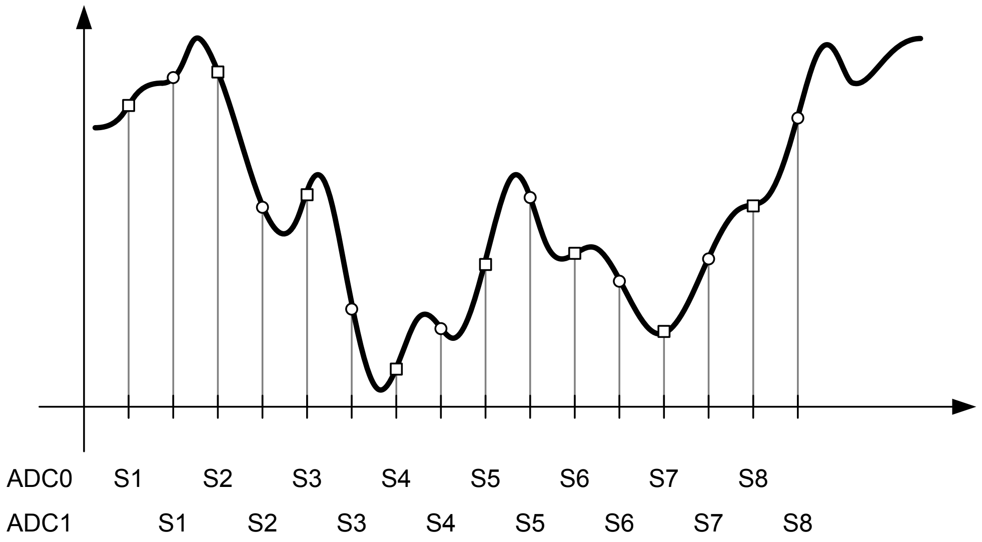
Using the **ADCSPC** register, ADC0 and ADC1 may provide a number of interesting applications:

- Coincident continuous sampling of different signals. The sample sequence steps run coincidently in both converters.
  - ADC Module 0, **ADCSPC** = 0x0, sampling `AIN0`
  - ADC Module 1, **ADCSPC** = 0x0, sampling `AIN1`

  Note: If two ADCs are configured to sample the same signal, a skew (phase lag) must be added to one of the ADC modules to prevent coincident sampling. Phase lag can be added by programming the `PHASE` field in the **ADCSPC** register.

- Skewed sampling of the same signal. The sample sequence steps are 0.5 µs out of phase with each other for 1 Msps. This configuration doubles the conversion bandwidth of a single input when software combines the results as shown in Figure 12-5 on page 737.
  - ADC Module 0, **ADCSPC** = 0x0, sampling `AIN0`
  - ADC Module 1, **ADCSPC** = 0x8, sampling `AIN0`

Figure 12-5. Skewed Sampling



12.3.2.6 Module Clocking

The module is clocked by a 16-MHz clock which can be sourced by a divided version of the PLL output, the PIOSC or an external source connected to MOSC (with the PLL in bypass mode). When the PLL is operating, the ADC clock is derived from the PLL ÷ 25 by default. However, the PIOSC can be used for the module clock using the **ADC Clock Configuration (ADCCC)** register. To use the PIOSC to clock the ADC, first power up the PLL and then enable the PIOSC in the `CS` bit field in the **ADCCC** register, then disable the PLL. When the PLL is bypassed, the module clock source clock attached to the MOSC must be 16 MHz unless the PIOSC is used for the clock source. To use the MOSC to clock the ADC, first power up the PLL and then enable the clock to the ADC module, then disable the PLL and switch to the MOSC for the system clock. The ADC module can continue to operate in Deep-Sleep mode if the PIOSC is the ADC module clock source.

The system clock must be at the same frequency or higher than the ADC clock. All ADC modules share the same clock source to facilitate the synchronization of data samples between conversion units, the selection and programming of which is provided by ADC0's **ADCCC** register. The ADC modules do not run at different conversion rates.

12.3.2.7 Busy Status

The `BUSY` bit of the **ADCACTSS** register is used to indicate when the ADC is busy with a current conversion. When there are no triggers pending which may start a new conversion in the immediate cycle or next few cycles, the `BUSY` bit reads as 0. Software must read the status of the `BUSY` bit as clear before disabling the ADC clock by writing to the **Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC)** register.

12.3.2.8 Dither Enable

The `DITHER` bit in the **ADCCTL** register is used to reduce random noise in ADC sampling and keep the ADC operation within the specified performance limits defined in Table 21-30 on page 1151. When taking multiple consecutive samples with the ADC Module, the `DITHER` bit should be enabled in the **ADCCTL** register along with hardware averaging in the **ADC Sample Averaging Control (ADCSAC)** register. The `DITHER` bit is disabled by default at reset.

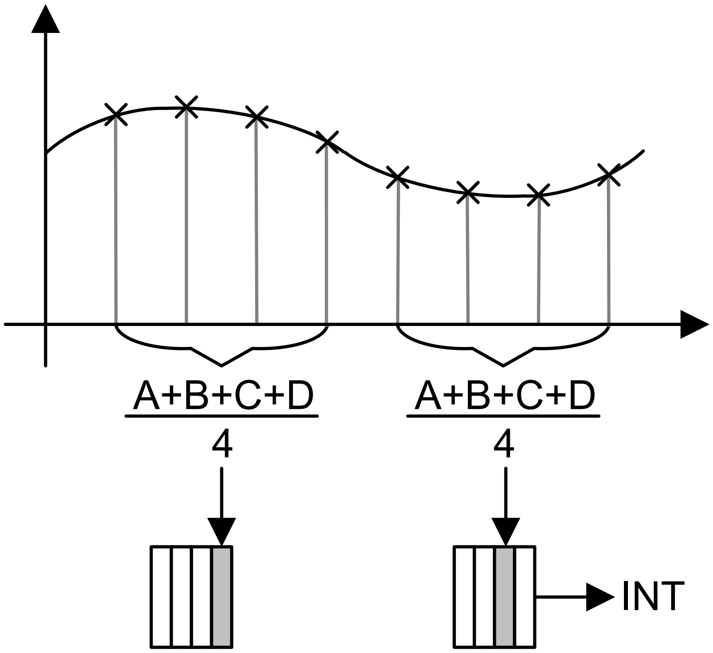
12.3.3 Hardware Sample Averaging Circuit

Higher precision results can be generated using the hardware averaging circuit, however, the improved results are at the cost of throughput. Up to 64 samples can be accumulated and averaged to form a single data entry in the sequencer FIFO. Throughput is decreased proportionally to the number of samples in the averaging calculation. For example, if the averaging circuit is configured to average 16 samples, the throughput is decreased by a factor of 16.

By default the averaging circuit is off, and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the **ADC Sample Averaging Control (ADCSAC)** register (see page 777). A single averaging circuit has been implemented, thus all input channels receive the same amount of averaging whether they are single-ended or differential.

Figure 12-6 shows an example in which the **ADCSAC** register is set to 0x2 for 4x hardware oversampling and the `IE1` bit is set for the sample sequence, resulting in an interrupt after the second averaged value is stored in the FIFO.

Figure 12-6. Sample Averaging Example

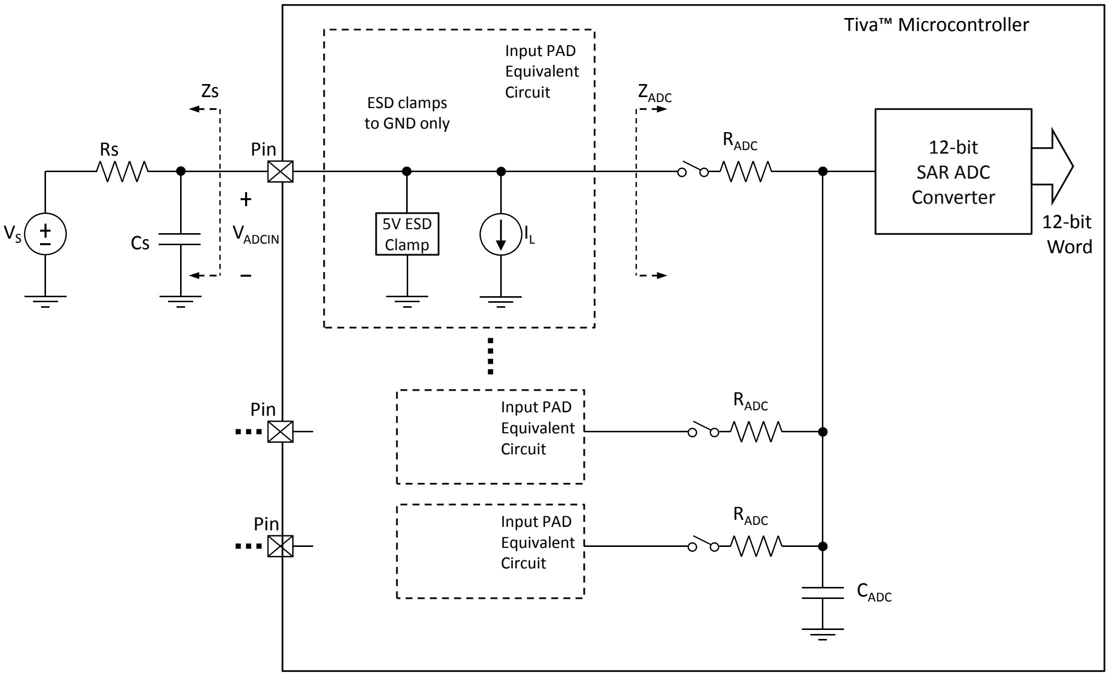


12.3.4 Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) module uses a Successive Approximation Register (SAR) architecture to deliver a 12-bit, low-power, high-precision conversion value. The successive approximation uses a switched capacitor array to perform the dual functions of sampling and holding the signal as well as providing the 12-bit DAC operation.

Figure 12-7 shows the ADC input equivalency diagram; for parameter values, see “Analog-to-Digital Converter (ADC)” on page 1151.

Figure 12-7. ADC Input Equivalency

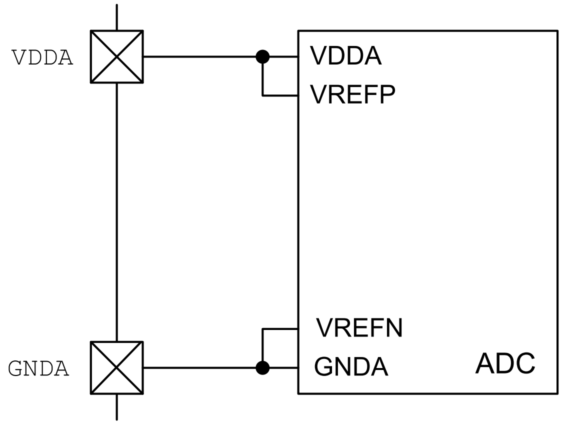


The ADC operates from both the 3.3-V analog and 1.2-V digital power supplies. The ADC clock can be configured to reduce power consumption when ADC conversions are not required (see “System Control” on page 215). The analog inputs are connected to the ADC through specially balanced input paths to minimize the distortion and cross-talk on the inputs. Detailed information on the ADC power supplies and analog inputs can be found in “Analog-to-Digital Converter (ADC)” on page 1151.

12.3.4.1 Voltage Reference

The ADC uses internal signals VREFP and VREFN as references to produce a conversion value from the selected analog input. VREFP is connected to `VDDA` and VREFN is connected to `GNDA`, as shown in Figure 12-8.

Figure 12-8. ADC Voltage Reference

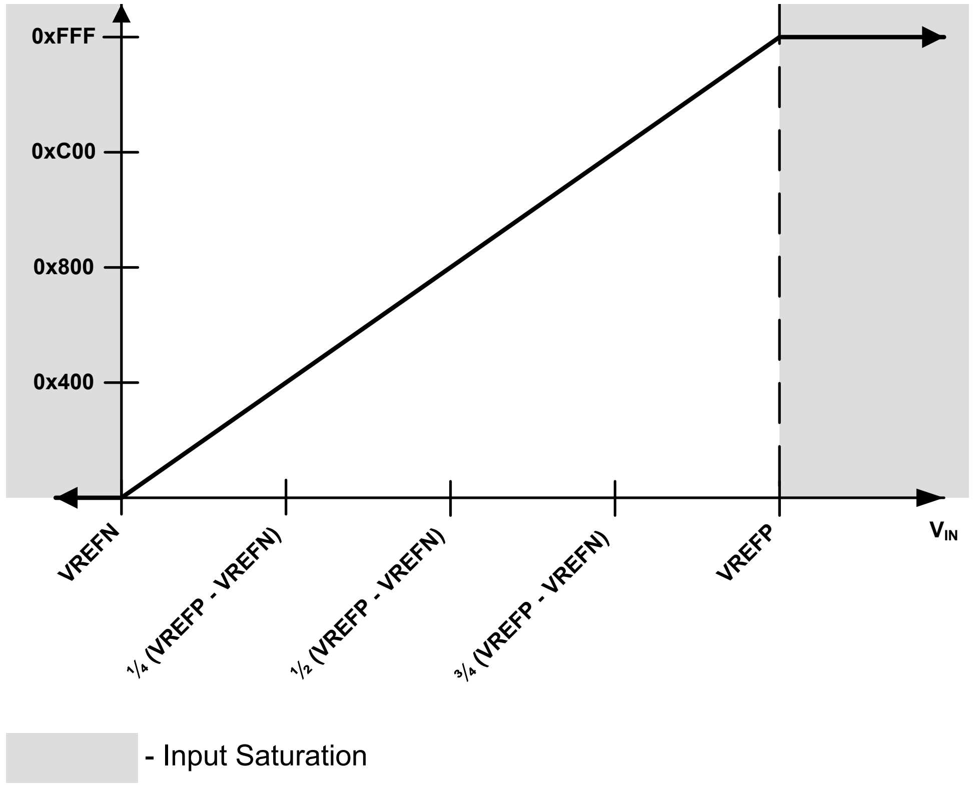


The range of this conversion value is from 0x000 to 0xFFF. In single-ended-input mode, the 0x000 value corresponds to the voltage level on VREFN; the 0xFFF value corresponds to the voltage level on VREFP. This configuration results in a resolution that can be calculated using the following equation:

```
mV per ADC code = (VREFP - VREFN) / 4096
```

While the analog input pads can handle voltages beyond this range, the analog input voltages must remain within the limits prescribed by Table 21-30 on page 1151 to produce accurate results. Figure 12-9 on page 742 shows the ADC conversion function of the analog inputs.

Figure 12-9. ADC Conversion Result



12.3.5 Differential Sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analog input channels. To enable differential sampling, software must set the Dn bit in the **ADCSSCTL0n** register in a step's configuration nibble.

When a sequence step is configured for differential sampling, the input pair to sample must be configured in the **ADCSSMUXn** register. Differential pair 0 samples analog inputs 0 and 1; differential pair 1 samples analog inputs 2 and 3; and so on (see Table 12-3 on page 742). The ADC does not support other differential pairings such as analog input 0 with analog input 3.

Table 12-3. 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

The voltage sampled in differential mode is the difference between the odd and even channels:

- Input Positive Voltage: VIN+ = V_{IN_EVEN} (even channel)
- Input Negative Voltage: VIN- = V_{IN_ODD} (odd channel)

The input differential voltage is defined as: VIN_D = VIN+ - VIN-, therefore:

- If VIN_D = 0, then the conversion result = 0x800
- If VIN_D > 0, then the conversion result > 0x800 (range is 0x800–0xFFF)
- If VIN_D < 0, then the conversion result < 0x800 (range is 0–0x800)

When using differential sampling, the following definitions are relevant:

- Input Common Mode Voltage: VIN_{CM} = (VIN+ + VIN-) / 2
- Reference Positive Voltage: VREFP
- Reference Negative Voltage: VREFN
- Reference Differential Voltage: $VREF_D$ = VREFP - VREFN
- Reference Common Mode Voltage: $VREF_{CM}$ = (VREFP + VREFN) / 2

The following conditions provide optimal results in differential mode:

- Both V_{IN_EVEN} and V_{IN_ODD} must be in the range of (VREFP to VREFN) for a valid conversion result
- The maximum possible differential input swing, or the maximum differential range, is: -$VREF_D$to +$VREF_D$, so the maximum peak-to-peak input differential signal is (+$VREF_D$ - -$VREF_D$) = 2 * $VREF_D$= 2 * (VREFP - VREFN)
- In order to take advantage of the maximum possible differential input swing, VIN_{CM} should be very close to $VREF_{CM}$, see Table 21-30 on page 1151.

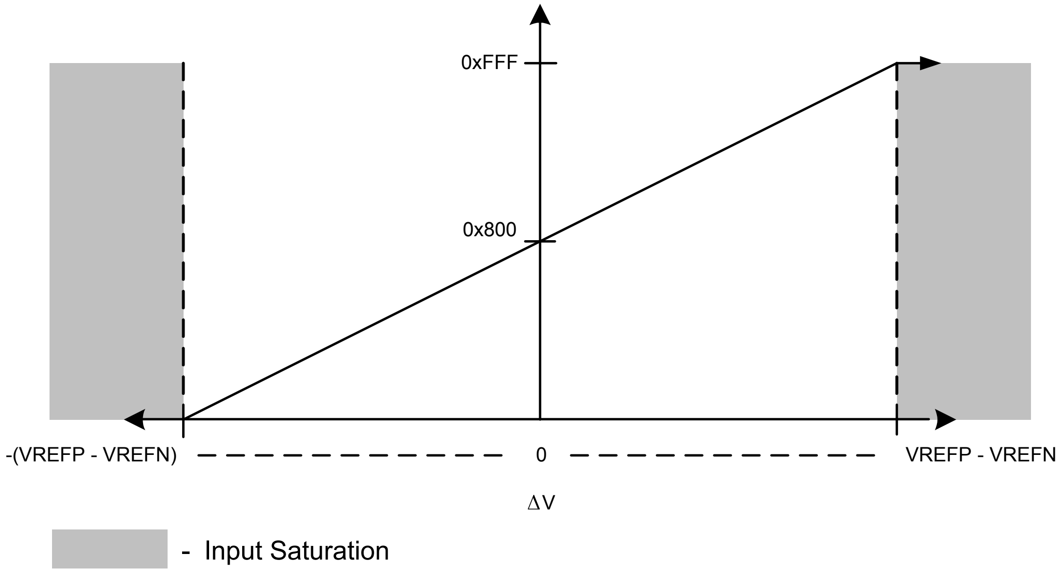
If VIN_{CM} is not equal to $VREF_{CM}$, the differential input signal may clip at either maximum or minimum voltage, because either single ended input can never be larger than VREFP or smaller than VREFN, and it is not possible to achieve full swing. Thus any difference in common mode between the input voltage and the reference voltage limits the differential dynamic range of the ADC.

Because the maximum peak-to-peak differential signal voltage is 2 * (VREFP - VREFN), the ADC codes are interpreted as:

```
mV per ADC code = (2 *(VREFP - VREFN)) / 4096
```

Figure 12-10 shows how the differential voltage, ∆V, is represented in ADC codes.

Figure 12-10. Differential Voltage Representation



12.3.6 Internal Temperature Sensor

The temperature sensor's primary purpose is to notify the system that the internal temperature is too high or low for reliable operation.

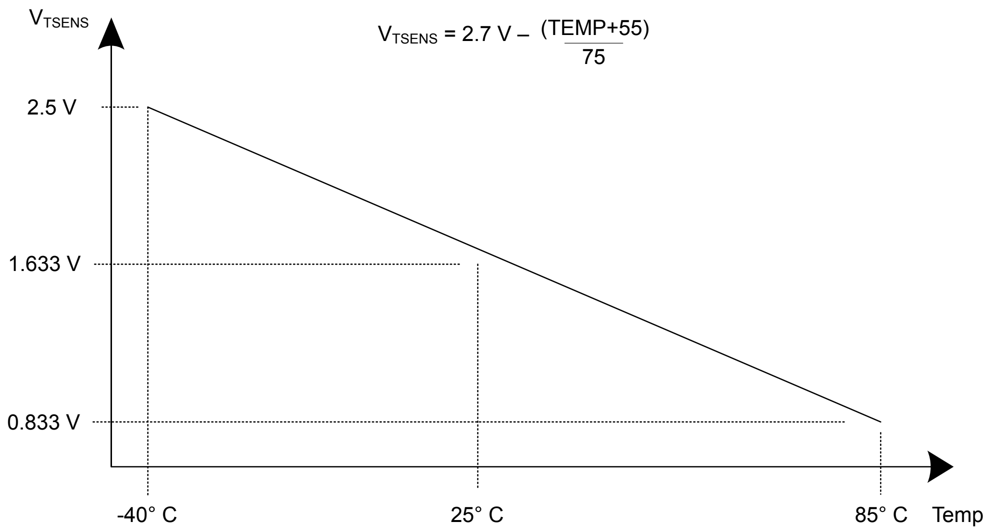
The temperature sensor does not have a separate enable, because it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC.

The internal temperature sensor converts a temperature measurement into a voltage. This voltage value, V_{TSENS}, is given by the following equation (where TEMP is the temperature in °C):

$$V_{TSENS} = 2.7 - ((TEMP + 55) / 75)$$

This relation is shown in Figure 12-11 on page 745.

Figure 12-11. Internal Temperature Sensor Characteristic



The temperature sensor reading can be sampled in a sample sequence by setting the TSn bit in the **ADCSSCTLn** register. The temperature reading from the temperature sensor can also be given as a function of the ADC value. The following formula calculates temperature (TEMP in ℃) based on the ADC reading (ADC_{CODE}, given as an unsigned decimal number from 0 to 4095) and the maximum ADC voltage range (VREFP - VREFN):

$$TEMP = 147.5 - ((75 * (VREFP - VREFN) \times ADC_{CODE}) / 4096)$$

12.3.7 Digital Comparator Unit

An ADC is commonly used to sample an external signal and to monitor its value to ensure that it remains in a given range. To automate this monitoring procedure and reduce the amount of processor overhead that is required, each module provides eight digital comparators.

Conversions from the ADC that are sent to the digital comparators are compared against the user programmable limits in the **ADC Digital Comparator Range (ADCDCCMPn)** registers. The ADC can be configured to generate an interrupt depending on whether the ADC is operating within the low, mid or high-band region configured in the ADCDCCMPn bit fields. The digital comparators four operational modes (Once, Always, Hysteresis Once, Hysteresis Always) can be additionally applied to the interrupt configuration.

12.3.7.1 Output Functions

ADC conversions can either be stored in the ADC Sample Sequence FIFOs or compared using the digital comparator resources as defined by the SnDCOP bits in the **ADC Sample Sequence n Operation (ADCSSOPn)** register. These selected ADC conversions are used by their respective digital comparator to monitor the external signal. Each comparator has two possible output functions: processor interrupts and triggers.

Each function has its own state machine to track the monitored signal. Even though the interrupt and trigger functions can be enabled individually or both at the same time, the same conversion data is used by each function to determine if the right conditions have been met to assert the associated output.

Interrupts

The digital comparator interrupt function is enabled by setting the `CIE` bit in the **ADC Digital Comparator Control (ADCDCCTLn)** register. This bit enables the interrupt function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, and the `DCONSSx` bit is set in the **ADCIM** register, an interrupt is sent to the interrupt controller.

Note: Only a single `DCONSSn` bit should be set at any given time. Setting more than one of these bits results in the `INRDC` bit from the **ADCRIS** register being masked, and no interrupt is generated on any of the sample sequencer interrupt lines. It is recommended that when interrupts are used, they are enabled on alternating samples or at the end of the sample sequence.

12.3.7.2 Operational Modes

Four operational modes are provided to support a broad range of applications and multiple possible signaling requirements: Always, Once, Hysteresis Always, and Hysteresis Once. The operational mode is selected using the `CIM` field in the **ADCDCCTLn** register.

Always Mode

In the Always operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria. The result is a string of assertions on the interrupt or trigger while the conversions are within the appropriate range.

Once Mode

In the Once operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria, and the previous ADC conversion value did not. The result is a single assertion of the interrupt or trigger when the conversions are within the appropriate range.

Hysteresis-Always Mode

The Hysteresis-Always operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Always mode, the associated interrupt or trigger is asserted in the following cases: 1) the ADC conversion value meets its comparison criteria or 2) a previous ADC conversion value has met the comparison criteria, and the hysteresis condition has not been cleared by entering the opposite region. The result is a string of assertions on the interrupt or trigger that continue until the opposite region is entered.

Hysteresis-Once Mode

The Hysteresis-Once operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Once mode, the associated interrupt or trigger is asserted only when the ADC conversion value meets its comparison criteria, the hysteresis condition is clear, and the previous ADC conversion did not meet the comparison criteria. The result is a single assertion on the interrupt or trigger.

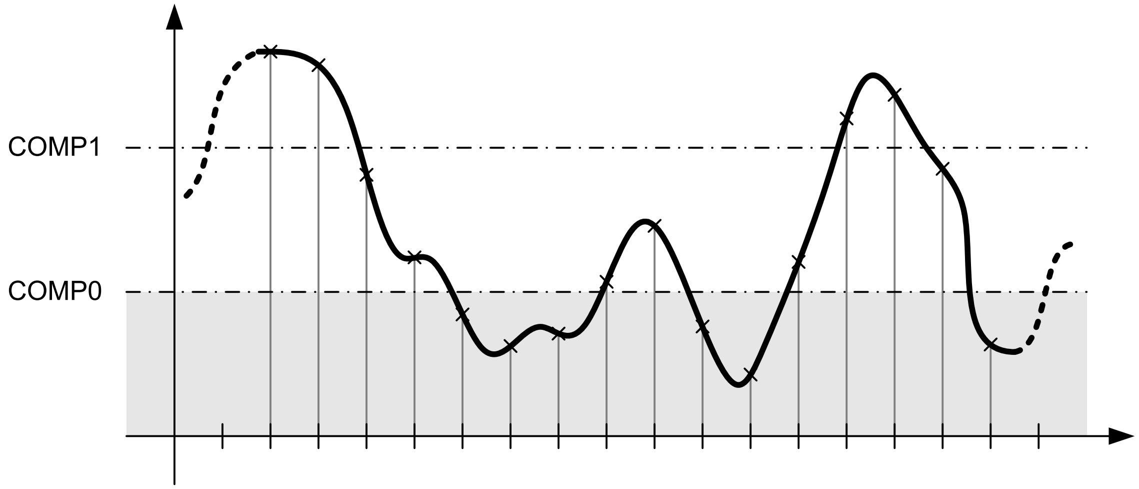
12.3.7.3 Function Ranges

The two comparison values, `COMP0` and `COMP1`, in the **ADC Digital Comparator Range (ADCDCCMPn)** register effectively break the conversion area into three distinct regions. These regions are referred to as the low-band (less than `COMP0`), mid-band (greater than `COMP0` but less than or equal to `COMP1`), and high-band (greater than or equal to `COMP1`) regions. `COMP0` and `COMP1` may be programmed to the same value, effectively creating two regions, but `COMP1` must always be greater than or equal to the value of `COMP0`. A `COMP1` value that is less than `COMP0` generates unpredictable results.

Low-Band Operation

To operate in the low-band region, the `CIC` field field in the **ADCDCCTLn** register must be programmed to 0x0. This setting causes interrupts or triggers to be generated in the low-band region as defined by the programmed operational mode. An example of the state of the interrupt/trigger signal in the low-band region for each of the operational modes is shown in Figure 12-12 on page 747. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is deasserted and a "1" indicates that the signal is asserted.

Figure 12-12. Low-Band Operation (CIC=0x0)

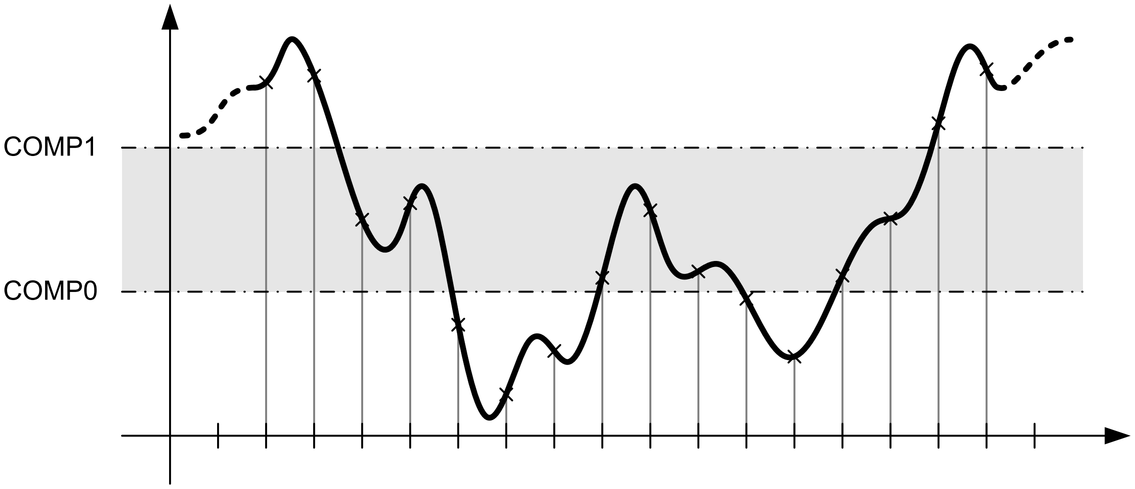


Always	–	0	0	0	0	1	1	1	0	0	1	1	0	0	0	0	1
Once	–	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
Hysteresis Always	–	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	1
Hysteresis Once	–	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

Mid-Band Operation

To operate in the mid-band region, the `CIC` field field in the **ADCDCCTLn** register must be programmed to 0x1. This setting causes interrupts or triggers to be generated in the mid-band region according the operation mode. Only the Always and Once operational modes are available in the mid-band region. An example of the state of the interrupt/trigger signal in the mid-band region for each of the allowed operational modes is shown in Figure 12-13 on page 748. Note that a "0" in a column following the operational mode name (Always or Once) indicates that the interrupt or trigger signal is deasserted and a "1" indicates that the signal is asserted.

Figure 12-13. Mid-Band Operation (CIC=0x1)

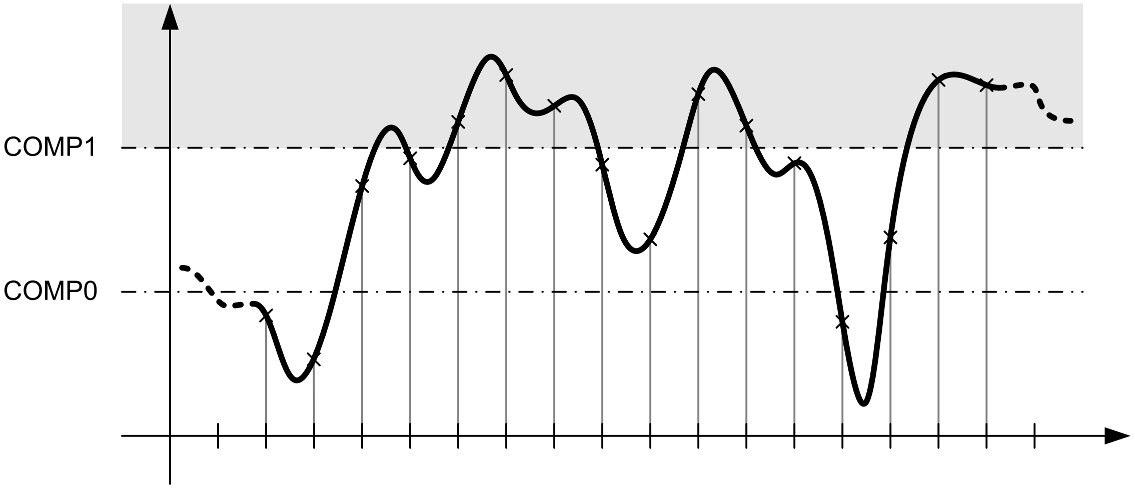


Always	–	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0	0
Once	–	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
Hysteresis Always	–	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Hysteresis Once	–	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

High-Band Operation

To operate in the high-band region, the `CIC` field field in the **ADCDCCTLn** register must be programmed to 0x3. This setting causes interrupts or triggers to be generated in the high-band region according the operation mode. An example of the state of the interrupt/trigger signal in the high-band region for each of the allowed operational modes is shown in Figure 12-14 on page 749. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is deasserted and a "1" indicates that the signal is asserted.

Figure 12-14. High-Band Operation (CIC=0x3)



Always	–	0	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1
Once	–	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
Hysteresis Always	–	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1
Hysteresis Once	–	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0

12.4 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and programmed to a supported crystal frequency in the **RCC** register (see page 241). Using unsupported frequencies can cause faulty operation in the ADC module.

12.4.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps: enabling the clock to the ADC, disabling the analog isolation circuit associated with all inputs that are to be used, and reconfiguring the sample sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock using the **RCGCADC** register (see page 330).
2. Enable the clock to the appropriate GPIO modules via the **RCGCGPIO** register (see page 319). To find out which GPIO ports to enable, refer to “Signal Description” on page 733.
3. Set the GPIO `AFSEL` bits for the ADC input pins (see page 603). To determine which GPIOs to configure, see Table 20-4 on page 1111.
4. Configure the `AINx` signals to be analog inputs by clearing the corresponding `DEN` bit in the **GPIO Digital Enable (GPIODEN)** register (see page 614).
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 619) in the associated GPIO block.

6. If required by the application, reconfigure the sample sequencer priorities in the **ADCSSPRI** register. The default configuration has Sample Sequencer 0 with the highest priority and Sample Sequencer 3 as the lowest priority.

12.4.2 Sample Sequencer Configuration

Configuration of the sample sequencers is slightly more complex than the module initialization because each sample sequencer is completely programmable.

The configuration for each sample sequencer should be as follows:

1. Ensure that the sample sequencer is disabled by clearing the corresponding ASENn bit in the **ADCACTSS** register. Programming of the sample sequencers is allowed without having them enabled. Disabling the sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.

2. Configure the trigger event for the sample sequencer in the **ADCEMUX** register.

3. For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUXn** register.

4. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTLn** register. When programming the last nibble, ensure that the END bit is set. Failure to set the END bit causes unpredictable behavior.

5. If interrupts are to be used, set the corresponding MASK bit in the **ADCIM** register.

6. Enable the sample sequencer logic by setting the corresponding ASENn bit in the **ADCACTSS** register.

12.5 Register Map

Table 12-4 on page 750 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 330). There must be a delay of 3 system clocks after the ADC module clock is enabled before any ADC module registers are accessed.

Table 12-4. ADC Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	RW	0x0000.0000	ADC Active Sample Sequencer	753
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	755
0x008	ADCIM	RW	0x0000.0000	ADC Interrupt Mask	757
0x00C	ADCISC	RW1C	0x0000.0000	ADC Interrupt Status and Clear	760
0x010	ADCOSTAT	RW1C	0x0000.0000	ADC Overflow Status	763
0x014	ADCEMUX	RW	0x0000.0000	ADC Event Multiplexer Select	765

Table 12-4. ADC Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x018	ADCUSTAT	RW1C	0x0000.0000	ADC Underflow Status	770
0x020	ADCSSPRI	RW	0x0000.3210	ADC Sample Sequencer Priority	771
0x024	ADCSPC	RW	0x0000.0000	ADC Sample Phase Control	773
0x028	ADCPSSI	RW	-	ADC Processor Sample Sequence Initiate	775
0x030	ADCSAC	RW	0x0000.0000	ADC Sample Averaging Control	777
0x034	ADCDCISC	RW1C	0x0000.0000	ADC Digital Comparator Interrupt Status and Clear	778
0x038	ADCCTL	RW	0x0000.0000	ADC Control	780
0x040	ADCSSMUX0	RW	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	781
0x044	ADCSSCTL0	RW	0x0000.0000	ADC Sample Sequence Control 0	783
0x048	ADCSSFIFO0	RO	-	ADC Sample Sequence Result FIFO 0	790
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	791
0x050	ADCSSOP0	RW	0x0000.0000	ADC Sample Sequence 0 Operation	793
0x054	ADCSSDC0	RW	0x0000.0000	ADC Sample Sequence 0 Digital Comparator Select	795
0x060	ADCSSMUX1	RW	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	797
0x064	ADCSSCTL1	RW	0x0000.0000	ADC Sample Sequence Control 1	798
0x068	ADCSSFIFO1	RO	-	ADC Sample Sequence Result FIFO 1	790
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	791
0x070	ADCSSOP1	RW	0x0000.0000	ADC Sample Sequence 1 Operation	802
0x074	ADCSSDC1	RW	0x0000.0000	ADC Sample Sequence 1 Digital Comparator Select	803
0x080	ADCSSMUX2	RW	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	797
0x084	ADCSSCTL2	RW	0x0000.0000	ADC Sample Sequence Control 2	798
0x088	ADCSSFIFO2	RO	-	ADC Sample Sequence Result FIFO 2	790
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	791
0x090	ADCSSOP2	RW	0x0000.0000	ADC Sample Sequence 2 Operation	802
0x094	ADCSSDC2	RW	0x0000.0000	ADC Sample Sequence 2 Digital Comparator Select	803
0x0A0	ADCSSMUX3	RW	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	805
0x0A4	ADCSSCTL3	RW	0x0000.0000	ADC Sample Sequence Control 3	806
0x0A8	ADCSSFIFO3	RO	-	ADC Sample Sequence Result FIFO 3	790
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	791
0x0B0	ADCSSOP3	RW	0x0000.0000	ADC Sample Sequence 3 Operation	808
0x0B4	ADCSSDC3	RW	0x0000.0000	ADC Sample Sequence 3 Digital Comparator Select	809
0xD00	ADCDCRIC	WO	0x0000.0000	ADC Digital Comparator Reset Initial Conditions	810

Table 12-4. ADC Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xE00	ADCDCCTL0	RW	0x0000.0000	ADC Digital Comparator Control 0	815
0xE04	ADCDCCTL1	RW	0x0000.0000	ADC Digital Comparator Control 1	815
0xE08	ADCDCCTL2	RW	0x0000.0000	ADC Digital Comparator Control 2	815
0xE0C	ADCDCCTL3	RW	0x0000.0000	ADC Digital Comparator Control 3	815
0xE10	ADCDCCTL4	RW	0x0000.0000	ADC Digital Comparator Control 4	815
0xE14	ADCDCCTL5	RW	0x0000.0000	ADC Digital Comparator Control 5	815
0xE18	ADCDCCTL6	RW	0x0000.0000	ADC Digital Comparator Control 6	815
0xE1C	ADCDCCTL7	RW	0x0000.0000	ADC Digital Comparator Control 7	815
0xE40	ADCDCCMP0	RW	0x0000.0000	ADC Digital Comparator Range 0	817
0xE44	ADCDCCMP1	RW	0x0000.0000	ADC Digital Comparator Range 1	817
0xE48	ADCDCCMP2	RW	0x0000.0000	ADC Digital Comparator Range 2	817
0xE4C	ADCDCCMP3	RW	0x0000.0000	ADC Digital Comparator Range 3	817
0xE50	ADCDCCMP4	RW	0x0000.0000	ADC Digital Comparator Range 4	817
0xE54	ADCDCCMP5	RW	0x0000.0000	ADC Digital Comparator Range 5	817
0xE58	ADCDCCMP6	RW	0x0000.0000	ADC Digital Comparator Range 6	817
0xE5C	ADCDCCMP7	RW	0x0000.0000	ADC Digital Comparator Range 7	817
0xFC0	ADCPP	RO	0x00B0.20C7	ADC Peripheral Properties	818
0xFC4	ADCPC	RW	0x0000.0007	ADC Peripheral Configuration	820
0xFC8	ADCCC	RW	0x0000.0000	ADC Clock Configuration	821

12.6 Register Descriptions

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

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

ADC Active Sample Sequencer (ADCACTSS)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x000
Type RW, reset 0x0000.0000

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

	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	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	BUSY	RO	0	ADC Busy Value Description 0 ADC is idle 1 ADC is busy
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	RW	0	ADC SS3 Enable Value Description 0 Sample Sequencer 3 is disabled. 1 Sample Sequencer 3 is enabled.
2	ASEN2	RW	0	ADC SS2 Enable Value Description 0 Sample Sequencer 2 is disabled. 1 Sample Sequencer 2 is enabled.
1	ASEN1	RW	0	ADC SS1 Enable Value Description 0 Sample Sequencer 1 is disabled. 1 Sample Sequencer 1 is enabled.

Bit/Field	Name	Type	Reset	Description
0	ASEN0	RW	0	ADC SS0 Enable Value Description 0 Sample Sequencer 0 is disabled. 1 Sample Sequencer 0 is enabled.

Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004

This register shows the status of the raw interrupt signal of each sample sequencer. These bits may be polled by software to look for interrupt conditions without sending the interrupts to the interrupt controller.

ADC Raw Interrupt Status (ADCRIS)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x004
Type RO, reset 0x0000.0000

	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 0 All bits in the **ADCDCISC** register are clear. 1 At least one bit in the **ADCDCISC** register is set, meaning that a digital comparator interrupt has occurred.
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INR3	RO	0	SS3 Raw Interrupt Status Value Description 0 An interrupt has not occurred. 1 A sample has completed conversion and the respective **ADCSSCTL3** `IEn` bit is set, enabling a raw interrupt. 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 0 An interrupt has not occurred. 1 A sample has completed conversion and the respective **ADCSSCTL2** `IEn` bit is set, enabling a raw interrupt. 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 0 An interrupt has not occurred. 1 A sample has completed conversion and the respective **ADCSSCTL1** `IEn` bit is set, enabling a raw interrupt. 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 0 An interrupt has not occurred. 1 A sample has completed conversion and the respective **ADCSSCTL0** `IEn` bit is set, enabling a raw interrupt. This bit is cleared by writing a 1 to the `IN0` bit in the **ADCISC** register.

Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the sample sequencer and digital comparator raw interrupt signals are sent to the interrupt controller. Each raw interrupt signal can be masked independently.

Note: Only a single `DCONSS`n bit should be set at any given time. Setting more than one of these bits results in the `INRDC` bit from the **ADCRIS** register being masked, and no interrupt is generated on any of the sample sequencer interrupt lines. It is recommended that when interrupts are used, they are enabled on alternating samples or at the end of the sample sequence.

ADC Interrupt Mask (ADCIM)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x008
Type RW, reset 0x0000.0000

	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	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												MASK3	MASK2	MASK1	MASK0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	DCONSS3	RW	0	Digital Comparator Interrupt on SS3 Value Description 0 The status of the digital comparators does not affect the SS3 interrupt status. 1 The raw interrupt signal from the digital comparators (`INRDC` bit in the **ADCRIS** register) is sent to the interrupt controller on the SS3 interrupt line.
18	DCONSS2	RW	0	Digital Comparator Interrupt on SS2 Value Description 0 The status of the digital comparators does not affect the SS2 interrupt status. 1 The raw interrupt signal from the digital comparators (`INRDC` bit in the **ADCRIS** register) is sent to the interrupt controller on the SS2 interrupt line.

Bit/Field	Name	Type	Reset	Description
17	DCONSS1	RW	0	Digital Comparator Interrupt on SS1 Value Description 0 The status of the digital comparators does not affect the SS1 interrupt status. 1 The raw interrupt signal from the digital comparators (`INRDC` bit in the **ADCRIS** register) is sent to the interrupt controller on the SS1 interrupt line.
16	DCONSS0	RW	0	Digital Comparator Interrupt on SS0 Value Description 0 The status of the digital comparators does not affect the SS0 interrupt status. 1 The raw interrupt signal from the digital comparators (`INRDC` bit in the **ADCRIS** register) is sent to the interrupt controller on the SS0 interrupt line.
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MASK3	RW	0	SS3 Interrupt Mask Value Description 0 The status of Sample Sequencer 3 does not affect the SS3 interrupt status. 1 The raw interrupt signal from Sample Sequencer 3 (**ADCRIS** register `INR3` bit) is sent to the interrupt controller.
2	MASK2	RW	0	SS2 Interrupt Mask Value Description 0 The status of Sample Sequencer 2 does not affect the SS2 interrupt status. 1 The raw interrupt signal from Sample Sequencer 2 (**ADCRIS** register `INR2` bit) is sent to the interrupt controller.
1	MASK1	RW	0	SS1 Interrupt Mask Value Description 0 The status of Sample Sequencer 1 does not affect the SS1 interrupt status. 1 The raw interrupt signal from Sample Sequencer 1 (**ADCRIS** register `INR1` bit) is sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
0	MASK0	RW	0	SS0 Interrupt Mask Value Description 0 The status of Sample Sequencer 0 does not affect the SS0 interrupt status. 1 The raw interrupt signal from Sample Sequencer 0 (**ADCRIS** register `INR0` bit) is sent to the interrupt controller.

Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

This register provides the mechanism for clearing sample sequencer interrupt conditions and shows the status of interrupts generated by the sample sequencers and the digital comparators which have been sent to the interrupt controller. When read, each bit field is the logical AND of the respective `INR` and `MASK` bits. Sample sequencer interrupts are cleared by writing a 1 to the corresponding bit position. Digital comparator interrupts are cleared by writing a 1 to the appropriate bits in the **ADCDCISC** register. If software is polling the **ADCRIS** instead of generating interrupts, the sample sequence `INRn` bits are still cleared via the **ADCISC** register, even if the `INn` bit is not set.

ADC Interrupt Status and Clear (ADCISC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x00C
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												DCINSS3	DCINSS2	DCINSS1	DCINSS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												IN3	IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW1C	RW1C	RW1C	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	DCINSS3	RO	0	Digital Comparator Interrupt Status on SS3 Value Description 0 No interrupt has occurred or the interrupt is masked. 1 Both the `INRDC` bit in the **ADCRIS** register and the `DCONSS3` bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 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 0 No interrupt has occurred or the interrupt is masked. 1 Both the `INRDC` bit in the **ADCRIS** register and the `DCONSS2` bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. This bit is cleared by writing a 1 to it. Clearing this bit also clears the `INRDC` bit in the **ADCRIS** register.

Bit/Field	Name	Type	Reset	Description
17	DCINSS1	RO	0	Digital Comparator Interrupt Status on SS1 Value Description 0 No interrupt has occurred or the interrupt is masked. 1 Both the INRDC bit in the **ADCRIS** register and the DCONSS1 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register.
16	DCINSS0	RO	0	Digital Comparator Interrupt Status on SS0 Value Description 0 No interrupt has occurred or the interrupt is masked. 1 Both the INRDC bit in the **ADCRIS** register and the DCONSS0 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register.
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IN3	RW1C	0	SS3 Interrupt Status and Clear Value Description 0 No interrupt has occurred or the interrupt is masked. 1 Both the INR3 bit in the **ADCRIS** register and the MASK3 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. This bit is cleared by writing a 1. Clearing this bit also clears the INR3 bit in the **ADCRIS** register.
2	IN2	RW1C	0	SS2 Interrupt Status and Clear Value Description 0 No interrupt has occurred or the interrupt is masked. 1 Both the INR2 bit in the **ADCRIS** register and the MASK2 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. This bit is cleared by writing a 1. Clearing this bit also clears the INR2 bit in the **ADCRIS** register.

Bit/Field	Name	Type	Reset	Description
1	IN1	RW1C	0	SS1 Interrupt Status and Clear Value Description 0 No interrupt has occurred or the interrupt is masked. 1 Both the `INR1` bit in the **ADCRIS** register and the `MASK1` bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. This bit is cleared by writing a 1. Clearing this bit also clears the `INR1` bit in the **ADCRIS** register.
0	IN0	RW1C	0	SS0 Interrupt Status and Clear Value Description 0 No interrupt has occurred or the interrupt is masked. 1 Both the `INR0` bit in the **ADCRIS** register and the `MASK0` bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. This bit is cleared by writing a 1. Clearing this bit also clears the `INR0` bit in the **ADCRIS** register.

Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010

This register indicates overflow conditions in the sample sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

ADC Overflow Status (ADCOSTAT)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x010
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	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
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OV3	RW1C	0	SS3 FIFO Overflow Value Description 0 The FIFO has not overflowed. 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. This bit is cleared by writing a 1.
2	OV2	RW1C	0	SS2 FIFO Overflow Value Description 0 The FIFO has not overflowed. 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. This bit is cleared by writing a 1.
1	OV1	RW1C	0	SS1 FIFO Overflow Value Description 0 The FIFO has not overflowed. 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. This bit is cleared by writing a 1.

Bit/Field	Name	Type	Reset	Description
0	OV0	RW1C	0	SS0 FIFO Overflow Value Description 0 The FIFO has not overflowed. 1 The FIFO for Sample Sequencer 0 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. This bit is cleared by writing a 1.

Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source.

ADC Event Multiplexer Select (ADCEMUX)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x014
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EM3				EM2				EM1				EM0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
15:12	EM3	RW	0x0	SS3 Trigger Select This field selects the trigger source for Sample Sequencer 3. The valid configurations for this field are:

Value	Event
0x0	Processor (default) The trigger is initiated by setting the `SSn` bit in the **ADCPSSI** register.
0x1	Analog Comparator 0 This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1095).
0x2	Analog Comparator 1 This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1095).
0x3	reserved
0x4	External (GPIO Pins) This trigger is connected to the GPIO interrupt for the corresponding GPIO (see "ADC Trigger Source" on page 587). **Note:** GPIOs that have `AINx` signals as alternate functions can be used to trigger the ADC. However, the pin cannot be used as both a GPIO and an analog input.
0x5	Timer In addition, the trigger must be enabled with the `TnOTE` bit in the **GPTMCTL** register (page 669).
0x6	reserved
0x7	reserved
0x8	reserved
0x9	reserved
0xA-0xE	reserved
0xF	Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
11:8	EM2	RW	0x0	SS2 Trigger Select This field selects the trigger source for Sample Sequencer 2. The valid configurations for this field are:

Value	Event
0x0	Processor (default) The trigger is initiated by setting the `SSn` bit in the **ADCPSSI** register.
0x1	Analog Comparator 0 This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1095).
0x2	Analog Comparator 1 This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1095).
0x3	reserved
0x4	External (GPIO Pins) This trigger is connected to the GPIO interrupt for the corresponding GPIO (see "ADC Trigger Source" on page 587). **Note:** GPIOs that have `AINx` signals as alternate functions can be used to trigger the ADC. However, the pin cannot be used as both a GPIO and an analog input.
0x5	Timer In addition, the trigger must be enabled with the `TnOTE` bit in the **GPTMCTL** register (page 669).
0x6	reserved
0x7	reserved
0x8	reserved
0x9	reserved
0xA-0xE	reserved
0xF	Always (continuously sample)

<table>
<tr><th>Bit/Field</th><th>Name</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>7:4</td><td>EM1</td><td>RW</td><td>0x0</td><td>SS1 Trigger Select
This field selects the trigger source for Sample Sequencer 1.
The valid configurations for this field are:
<table>
<tr><th>Value</th><th>Event</th></tr>
<tr><td>0x0</td><td>Processor (default)
The trigger is initiated by setting the SSn bit in the <b>ADCPSSI</b> register.</td></tr>
<tr><td>0x1</td><td>Analog Comparator 0
This trigger is configured by the <b>Analog Comparator Control 0 (ACCTL0)</b> register (page 1095).</td></tr>
<tr><td>0x2</td><td>Analog Comparator 1
This trigger is configured by the <b>Analog Comparator Control 1 (ACCTL1)</b> register (page 1095).</td></tr>
<tr><td>0x3</td><td>reserved</td></tr>
<tr><td>0x4</td><td>External (GPIO Pins)
This trigger is connected to the GPIO interrupt for the corresponding GPIO (see “ADC Trigger Source” on page 587).
<b>Note:</b> GPIOs that have AINx signals as alternate functions can be used to trigger the ADC. However, the pin cannot be used as both a GPIO and an analog input.</td></tr>
<tr><td>0x5</td><td>Timer
In addition, the trigger must be enabled with the TnOTE bit in the <b>GPTMCTL</b> register (page 669).</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>
</table>
</td></tr>
</table>

Bit/Field	Name	Type	Reset	Description
3:0	EM0	RW	0x0	SS0 Trigger Select This field selects the trigger source for Sample Sequencer 0 The valid configurations for this field are:

Value	Event
0x0	Processor (default) The trigger is initiated by setting the SSn bit in the **ADCPSSI** register.
0x1	Analog Comparator 0 This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1095).
0x2	Analog Comparator 1 This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1095).
0x3	reserved
0x4	External (GPIO Pins) This trigger is connected to the GPIO interrupt for the corresponding GPIO (see "ADC Trigger Source" on page 587). **Note:** GPIOs that have AINx signals as alternate functions can be used to trigger the ADC. However, the pin cannot be used as both a GPIO and an analog input.
0x5	Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 669).
0x6	reserved
0x7	reserved
0x8	reserved
0x9	reserved
0xA-0xE	reserved
0xF	Always (continuously sample)

Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018

This register indicates underflow conditions in the sample sequencer FIFOs. The corresponding underflow condition is cleared by writing a 1 to the relevant bit position.

ADC Underflow Status (ADCUSTAT)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x018
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												UV3	UV2	UV1	UV0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW1C	RW1C	RW1C	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	UV3	RW1C	0	SS3 FIFO Underflow The valid configurations for this field are shown below. This bit is cleared by writing a 1. Value Description 0 The FIFO has not underflowed. 1 The FIFO for the Sample Sequencer has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned.
2	UV2	RW1C	0	SS2 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.
1	UV1	RW1C	0	SS1 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.
0	UV0	RW1C	0	SS0 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.

Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the sample sequencers. Out of reset, Sequencer 0 has the highest priority, and Sequencer 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority for the ADC to operate properly.

ADC Sample Sequencer Priority (ADCSSPRI)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x020
Type RW, 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	RW	RW	RO	RO	RW	RW	RO	RO	RW	RW	RO	RO	RW	RW
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	RW	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	RW	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	RW	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	RW	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 Sample Phase Control (ADCSPC), offset 0x024

This register allows the ADC module to sample at one of 16 different discrete phases from 0.0° through 337.5°. For example, the sample rate could be effectively doubled by sampling a signal using one ADC module configured with the standard sample time and the second ADC module configured with a 180.0° phase lag.

Note: Care should be taken when the PHASE field is non-zero, as the resulting delay in sampling the AINx input may result in undesirable system consequences. The time from ADC trigger to sample is increased and could make the response time longer than anticipated. The added latency could have ramifications in the system design. Designers should carefully consider the impact of this delay.

ADC Sample Phase Control (ADCSPC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x024
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												PHASE			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
3:0	PHASE	RW	0x0	Phase Difference This field selects the sample phase difference from the standard sample time.

Value	Description
0x0	ADC sample lags by 0.0°
0x1	ADC sample lags by 22.5°
0x2	ADC sample lags by 45.0°
0x3	ADC sample lags by 67.5°
0x4	ADC sample lags by 90.0°
0x5	ADC sample lags by 112.5°
0x6	ADC sample lags by 135.0°
0x7	ADC sample lags by 157.5°
0x8	ADC sample lags by 180.0°
0x9	ADC sample lags by 202.5°
0xA	ADC sample lags by 225.0°
0xB	ADC sample lags by 247.5°
0xC	ADC sample lags by 270.0°
0xD	ADC sample lags by 292.5°
0xE	ADC sample lags by 315.0°
0xF	ADC sample lags by 337.5°

Register 10: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

This register provides a mechanism for application software to initiate sampling in the sample sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in **ADCSSPRI** dictate execution order.

This register also provides a means to configure and then initiate concurrent sampling on all ADC modules. To do this, the first ADC module should be configured. The **ADCPSSI** register for that module should then be written. The appropriate SS bits should be set along with the SYNCWAIT bit. Additional ADC modules should then be configured following the same procedure. Once the final ADC module is configured, its **ADCPSSI** register should be written with the appropriate SS bits set along with the GSYNC bit. All of the ADC modules then begin concurrent sampling according to their configuration.

ADC Processor Sample Sequence Initiate (ADCPSSI)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x028
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	GSYNC	reserved			SYNCWAIT	reserved										
Type	RW	RO	RO	RO	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												SS3	SS2	SS1	SS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31	GSYNC	RW	0	Global Synchronize Value Description 0 This bit is cleared once sampling has been initiated. 1 This bit initiates sampling in multiple ADC modules at the same time. Any ADC module that has been initialized by setting an SSn bit and the SYNCWAIT bit starts sampling once this bit is written.
30:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27	SYNCWAIT	RW	0	Synchronize Wait Value Description 0 Sampling begins when a sample sequence has been initiated. 1 This bit allows the sample sequences to be initiated, but delays sampling until the GSYNC bit is set.
26:4	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
3	SS3	WO	-	SS3 Initiate Value Description 0 No effect. 1 Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.
2	SS2	WO	-	SS2 Initiate Value Description 0 No effect. 1 Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.
1	SS1	WO	-	SS1 Initiate Value Description 0 No effect. 1 Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.
0	SS0	WO	-	SS0 Initiate Value Description 0 No effect. 1 Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.

Register 11: ADC Sample Averaging Control (ADCSAC), offset 0x030

This register controls the amount of hardware averaging applied to conversion results. The final conversion result stored in the FIFO is averaged from 2^{AVG} consecutive ADC samples at the specified ADC speed. If AVG is 0, the sample is passed directly through without any averaging. If AVG=6, then 64 consecutive ADC samples are averaged to generate one result in the sequencer FIFO. An AVG=7 provides unpredictable results.

ADC Sample Averaging Control (ADCSAC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x030
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													AVG		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	AVG	RW	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 12: ADC Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034

This register provides status and acknowledgement of digital comparator interrupts. One bit is provided for each comparator.

ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x034
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCINT7	RW1C	0	Digital Comparator 7 Interrupt Status and Clear Value Description 0 No interrupt. 1 Digital Comparator 7 has generated an interrupt. This bit is cleared by writing a 1.
6	DCINT6	RW1C	0	Digital Comparator 6 Interrupt Status and Clear Value Description 0 No interrupt. 1 Digital Comparator 6 has generated an interrupt. This bit is cleared by writing a 1.
5	DCINT5	RW1C	0	Digital Comparator 5 Interrupt Status and Clear Value Description 0 No interrupt. 1 Digital Comparator 5 has generated an interrupt. This bit is cleared by writing a 1.

Bit/Field	Name	Type	Reset	Description
4	DCINT4	RW1C	0	Digital Comparator 4 Interrupt Status and Clear Value Description 0 No interrupt. 1 Digital Comparator 4 has generated an interrupt. This bit is cleared by writing a 1.
3	DCINT3	RW1C	0	Digital Comparator 3 Interrupt Status and Clear Value Description 0 No interrupt. 1 Digital Comparator 3 has generated an interrupt. This bit is cleared by writing a 1.
2	DCINT2	RW1C	0	Digital Comparator 2 Interrupt Status and Clear Value Description 0 No interrupt. 1 Digital Comparator 2 has generated an interrupt. This bit is cleared by writing a 1.
1	DCINT1	RW1C	0	Digital Comparator 1 Interrupt Status and Clear Value Description 0 No interrupt. 1 Digital Comparator 1 has generated an interrupt. This bit is cleared by writing a 1.
0	DCINT0	RW1C	0	Digital Comparator 0 Interrupt Status and Clear Value Description 0 No interrupt. 1 Digital Comparator 0 has generated an interrupt. This bit is cleared by writing a 1.

Register 13: ADC Control (ADCCTL), offset 0x038

This register configures the voltage reference. Note that values set in this register apply to all ADC modules, it is not possible to set one module to use internal references and another to use external references.

ADC Control (ADCCTL)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x038
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									DITHER	reserved					VREF
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	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.
6	DITHER	RW	0	Dither Mode Enable

Value	Description
0	Dither mode disabled
1	Dither mode enabled

Bit/Field	Name	Type	Reset	Description
5:1	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	VREF	RW	0x0	Voltage Reference Select

Value	Description
0x0	VDDA and GNDA are the voltage references for all ADC modules.
0x1	Reserved

Register 14: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x040
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MUX7				MUX6				MUX5				MUX4			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MUX3				MUX2				MUX1				MUX0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:28	MUX7	RW	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	RW	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	RW	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	RW	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	RW	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	RW	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	RW	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	RW	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 15: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with a sample sequencer. When configuring a sample sequence, the `END` bit must be set for the final sample, whether it be after the first sample, eighth sample, or any sample in between. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Control 0 (ADCSSCTL0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x044
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	TS7	RW	0	8th Sample Temp Sensor Select

Value	Description
0	The input pin specified by the **ADCSSMUXn** register is read during the eighth sample of the sample sequence.
1	The temperature sensor is read during the eighth sample of the sample sequence.

Bit/Field	Name	Type	Reset	Description
30	IE7	RW	0	8th Sample Interrupt Enable

Value	Description
0	The raw interrupt is not asserted to the interrupt controller.
1	The raw interrupt signal (`INR0` bit) is asserted at the end of the eighth sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.

It is legal to have multiple samples within a sequence generate interrupts.

Bit/Field	Name	Type	Reset	Description
29	END7	RW	0	8th Sample is End of Sequence

Value	Description
0	Another sample in the sequence is the final sample.
1	The eighth sample is the last sample of the sequence.

It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.

Bit/Field	Name	Type	Reset	Description
28	D7	RW	0	8th Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS7` bit is set.
27	TS6	RW	0	7th Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the seventh sample of the sample sequence. 1 The temperature sensor is read during the seventh sample of the sample sequence.
26	IE6	RW	0	7th Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (`INR0` bit) is asserted at the end of the seventh sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
25	END6	RW	0	7th Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The seventh sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.
24	D6	RW	0	7th Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS6` bit is set.

Bit/Field	Name	Type	Reset	Description
23	TS5	RW	0	6th Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the sixth sample of the sample sequence. 1 The temperature sensor is read during the sixth sample of the sample sequence.
22	IE5	RW	0	6th Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (`INR0` bit) is asserted at the end of the sixth sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
21	END5	RW	0	6th Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The sixth sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.
20	D5	RW	0	6th Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS5` bit is set.
19	TS4	RW	0	5th Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the fifth sample of the sample sequence. 1 The temperature sensor is read during the fifth sample of the sample sequence.

Bit/Field	Name	Type	Reset	Description
18	IE4	RW	0	5th Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (INR0 bit) is asserted at the end of the fifth sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
17	END4	RW	0	5th Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The fifth sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.
16	D4	RW	0	5th Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the TS4 bit is set.
15	TS3	RW	0	4th Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the fourth sample of the sample sequence. 1 The temperature sensor is read during the fourth sample of the sample sequence.
14	IE3	RW	0	4th Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (INR0 bit) is asserted at the end of the fourth sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.

Bit/Field	Name	Type	Reset	Description
13	END3	RW	0	4th Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The fourth sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.
12	D3	RW	0	4th Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS3` bit is set.
11	TS2	RW	0	3rd Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the third sample of the sample sequence. 1 The temperature sensor is read during the third sample of the sample sequence.
10	IE2	RW	0	3rd Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (`INR0` bit) is asserted at the end of the third sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
9	END2	RW	0	3rd Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The third sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.

Bit/Field	Name	Type	Reset	Description
8	D2	RW	0	3rd Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS2` bit is set.
7	TS1	RW	0	2nd Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the second sample of the sample sequence. 1 The temperature sensor is read during the second sample of the sample sequence.
6	IE1	RW	0	2nd Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (`INR0` bit) is asserted at the end of the second sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
5	END1	RW	0	2nd Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The second sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.
4	D1	RW	0	2nd Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS1` bit is set.

Bit/Field	Name	Type	Reset	Description
3	TS0	RW	0	1st Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the first sample of the sample sequence. 1 The temperature sensor is read during the first sample of the sample sequence.
2	IE0	RW	0	1st Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (`INR0` bit) is asserted at the end of the first sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
1	END0	RW	0	1st Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The first sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.
0	D0	RW	0	1st Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS0` bit is set.

Register 16: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048

Register 17: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068

Register 18: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088

Register 19: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8

Important: This register is read-sensitive. See the register description for details.

This register contains the conversion results for samples collected with the sample sequencer (the **ADCSSFIFO0** register is used for Sample Sequencer 0, **ADCSSFIFO1** for Sequencer 1, **ADCSSFIFO2** for Sequencer 2, and **ADCSSFIFO3** for Sequencer 3). Reads of this register return conversion result data in the order sample 0, sample 1, and so on, until the FIFO is empty. If the FIFO is not properly handled by software, overflow and underflow conditions are registered in the **ADCOSTAT** and **ADCUSTAT** registers.

ADC Sample Sequence Result FIFO n (ADCSSFIFOn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x048
Type RO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	RO	RO	RO	RO	RO	RO	RO	RO
Reset	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.
11:0	DATA	RO	-	Conversion Result Data

Register 20: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C

Register 21: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C

Register 22: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C

Register 23: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC

This register provides a window into the sample sequencer, providing full/empty status information as well as the positions of the head and tail pointers. The reset value of 0x100 indicates an empty FIFO with the head and tail pointers both pointing to index 0. The **ADCSSFSTAT0** register provides status on FIFO0, which has 8 entries; **ADCSSFSTAT1** on FIFO1, which has 4 entries; **ADCSSFSTAT2** on FIFO2, which has 4 entries; and **ADCSSFSTAT3** on FIFO3 which has a single entry.

ADC Sample Sequence FIFO n Status (ADCSSFSTATn)

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
0	The FIFO is not currently full.
1	The FIFO is currently full.

Bit/Field	Name	Type	Reset	Description
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
0	The FIFO is not currently empty.
1	The FIFO is 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. Valid values are 0x0-0x7 for FIFO0; 0x0-0x3 for FIFO1 and FIFO2; and 0x0 for FIFO3.
3:0	TPTR	RO	0x0	FIFO Tail Pointer This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read. Valid values are 0x0-0x7 for FIFO0; 0x0-0x3 for FIFO1 and FIFO2; and 0x0 for FIFO3.

Register 24: ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050

This register determines whether the sample from the given conversion on Sample Sequence 0 is saved in the Sample Sequence FIFO0 or sent to the digital comparator unit.

ADC Sample Sequence 0 Operation (ADCSSOP0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x050
Type RW, 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	RW	RO	RO	RO	RW	RO	RO	RO	RW	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			S3DCOP	reserved			S2DCOP	reserved			S1DCOP	reserved			S0DCOP
Type	RO	RO	RO	RW	RO	RO	RO	RW	RO	RO	RO	RW	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	S7DCOP	RW	0	Sample 7 Digital Comparator Operation

Value	Description
0	The eighth sample is saved in Sample Sequence FIFO0.
1	The eighth sample is sent to the digital comparator unit specified by the `S7DCSEL` bit in the **ADCSSDC0** register, and the value is not written to the FIFO.

Bit/Field	Name	Type	Reset	Description
27:25	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	S6DCOP	RW	0	Sample 6 Digital Comparator Operation Same definition as `S7DCOP` but used during the seventh sample.
23:21	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20	S5DCOP	RW	0	Sample 5 Digital Comparator Operation Same definition as `S7DCOP` but used during the sixth sample.
19:17	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	S4DCOP	RW	0	Sample 4 Digital Comparator Operation Same definition as `S7DCOP` but used during the fifth sample.
15:13	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
12	S3DCOP	RW	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	RW	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	RW	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	RW	0	Sample 0 Digital Comparator Operation Same definition as S7DCOP but used during the first sample.

Register 25: ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 0, if the corresponding SnDCOP bit in the **ADCSSOP0** register is set.

ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x054
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	S7DCSEL				S6DCSEL				S5DCSEL				S4DCSEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:28	S7DCSEL	RW	0x0	Sample 7 Digital Comparator Select 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. **Note:** Values not listed are reserved. Value Description 0x0 Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCDCCTL0**) 0x1 Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCDCCTL1**) 0x2 Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCDCCTL2**) 0x3 Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCDCCTL3**) 0x4 Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCDCCTL4**) 0x5 Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCDCCTL5**) 0x6 Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCDCCTL6**) 0x7 Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCDCCTL7**)
27:24	S6DCSEL	RW	0x0	Sample 6 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the seventh sample.
23:20	S5DCSEL	RW	0x0	Sample 5 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the sixth sample.
19:16	S4DCSEL	RW	0x0	Sample 4 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the fifth sample.
15:12	S3DCSEL	RW	0x0	Sample 3 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the fourth sample.

Bit/Field	Name	Type	Reset	Description
11:8	S2DCSEL	RW	0x0	Sample 2 Digital Comparator Select This field has the same encodings as `S7DCSEL` but is used during the third sample.
7:4	S1DCSEL	RW	0x0	Sample 1 Digital Comparator Select This field has the same encodings as `S7DCSEL` but is used during the second sample.
3:0	S0DCSEL	RW	0x0	Sample 0 Digital Comparator Select This field has the same encodings as `S7DCSEL` but is used during the first sample.

Register 26: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

Register 27: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1 or 2. These registers are 16 bits wide and contain information for four possible samples. See the **ADCSSMUX0** register on page 781 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 n (ADCSSMUXn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x060
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MUX3				MUX2				MUX1				MUX0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	MUX3	RW	0x0	4th Sample Input Select
11:8	MUX2	RW	0x0	3rd Sample Input Select
7:4	MUX1	RW	0x0	2nd Sample Input Select
3:0	MUX0	RW	0x0	1st Sample Input Select

Register 28: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

Register 29: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the `END` bit must be set for the final sample, whether it be after the first sample, fourth sample, or any sample in between. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSCTL0** register on page 783 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

ADC Sample Sequence Control n (ADCSSCTLn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x064
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	TS3	RW	0	4th Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the fourth sample of the sample sequence. 1 The temperature sensor is read during the fourth sample of the sample sequence.
14	IE3	RW	0	4th Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (`INR0` bit) is asserted at the end of the fourth sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.

Bit/Field	Name	Type	Reset	Description
13	END3	RW	0	4th Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The fourth sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.
12	D3	RW	0	4th Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS3` bit is set.
11	TS2	RW	0	3rd Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the third sample of the sample sequence. 1 The temperature sensor is read during the third sample of the sample sequence.
10	IE2	RW	0	3rd Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (`INR0` bit) is asserted at the end of the third sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
9	END2	RW	0	3rd Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The third sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.

Bit/Field	Name	Type	Reset	Description
8	D2	RW	0	3rd Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS2` bit is set.
7	TS1	RW	0	2nd Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the second sample of the sample sequence. 1 The temperature sensor is read during the second sample of the sample sequence.
6	IE1	RW	0	2nd Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (`INR0` bit) is asserted at the end of the second sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
5	END1	RW	0	2nd Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The second sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.
4	D1	RW	0	2nd Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS1` bit is set.

Bit/Field	Name	Type	Reset	Description
3	TS0	RW	0	1st Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the first sample of the sample sequence. 1 The temperature sensor is read during the first sample of the sample sequence.
2	IE0	RW	0	1st Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (`INR0` bit) is asserted at the end of the first sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
1	END0	RW	0	1st Sample is End of Sequence Value Description 0 Another sample in the sequence is the final sample. 1 The first sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.
0	D0	RW	0	1st Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the `TS0` bit is set.

Register 30: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070

Register 31: ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090

This register determines whether the sample from the given conversion on Sample Sequence n is saved in the Sample Sequence n FIFO or sent to the digital comparator unit. The **ADCSSOP1** register controls Sample Sequencer 1 and the **ADCSSOP2** register controls Sample Sequencer 2.

ADC Sample Sequence n Operation (ADCSSOPn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x070
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			S3DCOP	reserved			S2DCOP	reserved			S1DCOP	reserved			S0DCOP
Type	RO	RO	RO	RW	RO	RO	RO	RW	RO	RO	RO	RW	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	S3DCOP	RW	0	Sample 3 Digital Comparator Operation Value Description 0 The fourth sample is saved in Sample Sequence FIFOn. 1 The fourth sample is sent to the digital comparator unit specified by the S3DCSEL bit in the **ADCSSDC0n** register, and the value is not written to the FIFO.
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	S2DCOP	RW	0	Sample 2 Digital Comparator Operation Same definition as S3DCOP but used during the third sample.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	S1DCOP	RW	0	Sample 1 Digital Comparator Operation Same definition as S3DCOP but used during the second sample.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	RW	0	Sample 0 Digital Comparator Operation Same definition as S3DCOP but used during the first sample.

Register 32: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074

Register 33: ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094

These registers determine which digital comparator receives the sample from the given conversion on Sample Sequence n if the corresponding `SnDCOP` bit in the **ADCSSOPn** register is set. The **ADCSSDC1** register controls the selection for Sample Sequencer 1 and the **ADCSSDC2** register controls the selection for Sample Sequencer 2.

ADC Sample Sequence n Digital Comparator Select (ADCSSDCn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x074
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	S3DCSEL	RW	0x0	Sample 3 Digital Comparator Select 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. **Note:** Values not listed are reserved. Value Description 0x0 Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCCCTL0**) 0x1 Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCCCTL1**) 0x2 Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCCCTL2**) 0x3 Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCCCTL3**) 0x4 Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCCCTL4**) 0x5 Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCCCTL5**) 0x6 Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCCCTL6**) 0x7 Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCCCTL7**)
11:8	S2DCSEL	RW	0x0	Sample 2 Digital Comparator Select This field has the same encodings as `S3DCSEL` but is used during the third sample.

Bit/Field	Name	Type	Reset	Description
7:4	S1DCSEL	RW	0x0	Sample 1 Digital Comparator Select This field has the same encodings as `S3DCSEL` but is used during the second sample.
3:0	S0DCSEL	RW	0x0	Sample 0 Digital Comparator Select This field has the same encodings as `S3DCSEL` but is used during the first sample.

Register 34: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register defines the analog input configuration for the sample executed with Sample Sequencer 3. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 781 for detailed bit descriptions.

ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0A0
Type RW, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	MUX0	RW	0	1st Sample Input Select

Register 35: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for a sample executed with Sample Sequencer 3. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSCTL0** register on page 783 for detailed bit descriptions.

Note: When configuring a sample sequence in this register, the `END0` bit must be set.

ADC Sample Sequence Control 3 (ADCSSCTL3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0A4
Type RW, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	RW	0	1st Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the first sample of the sample sequence. 1 The temperature sensor is read during the first sample of the sample sequence.
2	IE0	RW	0	Sample Interrupt Enable Value Description 0 The raw interrupt is not asserted to the interrupt controller. 1 The raw interrupt signal (`INR0` bit) is asserted at the end of this sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
1	END0	RW	0	End of Sequence This bit must be set before initiating a single sample sequence. Value Description 0 Sampling and conversion continues. 1 This is the end of sequence.

Bit/Field	Name	Type	Reset	Description
0	D0	RW	0	Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set.

Register 36: ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0

This register determines whether the sample from the given conversion on Sample Sequence 3 is saved in the Sample Sequence 3 FIFO or sent to the digital comparator unit.

ADC Sample Sequence 3 Operation (ADCSSOP3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0B0
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															S0DCOP
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	RW	0	Sample 0 Digital Comparator Operation

Value	Description
0	The sample is saved in Sample Sequence FIFO3.
1	The sample is sent to the digital comparator unit specified by the `S0DCSEL` bit in the **ADCSSDC03** register, and the value is not written to the FIFO.

Register 37: ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 3 if the corresponding `SnDCOP` bit in the **ADCSSOP3** register is set.

ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0B4
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												S0DCSEL			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	S0DCSEL	RW	0x0	Sample 0 Digital Comparator Select

When the `S0DCOP` bit in the **ADCSSOP3** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the sample from Sample Sequencer 3.

Note: Values not listed are reserved.

Value	Description
0x0	Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCCCTL0**)
0x1	Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCCCTL1**)
0x2	Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCCCTL2**)
0x3	Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCCCTL3**)
0x4	Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCCCTL4**)
0x5	Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCCCTL5**)
0x6	Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCCCTL6**)
0x7	Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCCCTL7**)

Register 38: ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00

This register provides the ability to reset any of the digital comparator interrupt or trigger functions back to their initial conditions. Resetting these functions ensures that the data that is being used by the interrupt and trigger functions in the digital comparator unit is not stale.

ADC Digital Comparator Reset Initial Conditions (ADCDCRIC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xD00
Type WO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								DCTRIG7	DCTRIG6	DCTRIG5	DCTRIG4	DCTRIG3	DCTRIG2	DCTRIG1	DCTRIG0
Type	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	DCTRIG7	WO	0	Digital Comparator Trigger 7 Value Description 0 No effect. 1 Resets the Digital Comparator 7 trigger unit to its initial conditions. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. After setting this bit, software should wait until the bit clears before continuing.
22	DCTRIG6	WO	0	Digital Comparator Trigger 6 Value Description 0 No effect. 1 Resets the Digital Comparator 6 trigger unit to its initial conditions. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
21	DCTRIG5	WO	0	Digital Comparator Trigger 5 Value Description 0 No effect. 1 Resets the Digital Comparator 5 trigger unit to its initial conditions. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
20	DCTRIG4	WO	0	Digital Comparator Trigger 4 Value Description 0 No effect. 1 Resets the Digital Comparator 4 trigger unit to its initial conditions. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
19	DCTRIG3	WO	0	Digital Comparator Trigger 3 Value Description 0 No effect. 1 Resets the Digital Comparator 3 trigger unit to its initial conditions. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
18	DCTRIG2	WO	0	Digital Comparator Trigger 2 Value Description 0 No effect. 1 Resets the Digital Comparator 2 trigger unit to its initial conditions. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
17	DCTRIG1	WO	0	Digital Comparator Trigger 1 Value Description 0 No effect. 1 Resets the Digital Comparator 1 trigger unit to its initial conditions. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
16	DCTRIG0	WO	0	Digital Comparator Trigger 0 Value Description 0 No effect. 1 Resets the Digital Comparator 0 trigger unit to its initial conditions. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCINT7	WO	0	Digital Comparator Interrupt 7 Value Description 0 No effect. 1 Resets the Digital Comparator 7 interrupt unit to its initial conditions. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
6	DCINT6	WO	0	Digital Comparator Interrupt 6 Value Description 0 No effect. 1 Resets the Digital Comparator 6 interrupt unit to its initial conditions. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
5	DCINT5	WO	0	Digital Comparator Interrupt 5 Value Description 0 No effect. 1 Resets the Digital Comparator 5 interrupt unit to its initial conditions. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
4	DCINT4	WO	0	Digital Comparator Interrupt 4 Value Description 0 No effect. 1 Resets the Digital Comparator 4 interrupt unit to its initial conditions. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
3	DCINT3	WO	0	Digital Comparator Interrupt 3 Value Description 0 No effect. 1 Resets the Digital Comparator 3 interrupt unit to its initial conditions. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
2	DCINT2	WO	0	Digital Comparator Interrupt 2 Value Description 0 No effect. 1 Resets the Digital Comparator 2 interrupt unit to its initial conditions. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
1	DCINT1	WO	0	Digital Comparator Interrupt 1 Value Description 0 No effect. 1 Resets the Digital Comparator 1 interrupt unit to its initial conditions. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
0	DCINT0	WO	0	Digital Comparator Interrupt 0 Value Description 0 No effect. 1 Resets the Digital Comparator 0 interrupt unit to its initial conditions. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Register 39: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00

Register 40: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04

Register 41: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08

Register 42: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C

Register 43: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10

Register 44: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14

Register 45: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18

Register 46: ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C

This register provides the comparison encodings that generate an interrupt.

ADC Digital Comparator Control n (ADCDCCTLn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xE00
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											CIE	CIC		CIM	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	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	CIE	RW	0	Comparison Interrupt Enable

Value	Description
0	Disables the comparison interrupt. ADC conversion data has no effect on interrupt generation.
1	Enables the comparison interrupt. The ADC conversion data is used to determine if an interrupt should be generated according to the programming of the `CIC` and `CIM` fields.

Bit/Field	Name	Type	Reset	Description
3:2	CIC	RW	0x0	Comparison Interrupt Condition 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. Value Description 0x0 Low Band ADC Data < `COMP0` ≤ `COMP1` 0x1 Mid Band `COMP0` ≤ ADC Data < `COMP1` 0x2 reserved 0x3 High Band `COMP0` < `COMP1` ≤ ADC Data
1:0	CIM	RW	0x0	Comparison Interrupt Mode This field specifies the mode by which the interrupt comparison is made. Value Description 0x0 Always This mode generates an interrupt every time the ADC conversion data falls within the selected operational region. 0x1 Once This mode generates an interrupt the first time that the ADC conversion data enters the selected operational region. 0x2 Hysteresis Always 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. 0x3 Hysteresis Once This mode generates an interrupt the first time that the ADC conversion data falls within the selected operational region. No additional interrupts are generated until the hysteresis condition is cleared by entering the opposite operational region.

Register 47: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40

Register 48: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44

Register 49: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48

Register 50: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C

Register 51: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50

Register 52: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54

Register 53: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58

Register 54: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C

This register defines the comparison values that are used to determine if the ADC conversion data falls in the appropriate operating region.

Note: The value in the `COMP1` field must be greater than or equal to the value in the `COMP0` field or unexpected results can occur.

ADC Digital Comparator Range n (ADCDCCMPn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xE40
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				COMP1											
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				COMP0											
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27:16	COMP1	RW	0x000	Compare 1 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`.
15:12	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:0	COMP0	RW	0x000	Compare 0 The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the low-band region.

Register 55: ADC Peripheral Properties (ADCPP), offset 0xFC0

The **ADCPP** register provides information regarding the properties of the ADC module.

ADC Peripheral Properties (ADCPP)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xFC0
Type RO, reset 0x00B0.20C7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								TS	RSL					TYPE	
Type	RO	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	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DC						CH						MSR			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	1	0	0	0	0	0	1	1	0	0	0	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	TS	RO	0x1	Temperature Sensor Value Description 0 The ADC module does not have a temperature sensor. 1 The ADC module has a temperature sensor. This field provides the similar information as the legacy **DC1** register `TEMPSNS` bit.
22:18	RSL	RO	0xC	Resolution This field specifies the maximum number of binary bits used to represent the converted sample. The field is encoded as a binary value, in the range of 0 to 32 bits.
17:16	TYPE	RO	0x0	ADC Architecture Value Description 0x0 SAR 0x1 - 0x3 Reserved
15:10	DC	RO	0x8	Digital Comparator Count This field specifies the number of ADC digital comparators available to the converter. The field is encoded as a binary value, in the range of 0 to 63. This field provides similar information to the legacy **DC9** register `ADCnDCn` bits.

Bit/Field	Name	Type	Reset	Description
9:4	CH	RO	0xC	ADC Channel Count This field specifies the number of ADC input channels available to the converter. This field is encoded as a binary value, in the range of 0 to 63. This field provides similar information to the legacy **DC3** and **DC8** register ADCnAINn bits.
3:0	MSR	RO	0x7	Maximum ADC Sample Rate This field specifies the maximum number of ADC conversions per second. The MSR field is encoded as follows:

Value	Description
0x0	Reserved
0x1	125 ksps
0x2	Reserved
0x3	250 ksps
0x4	Reserved
0x5	500 ksps
0x6	Reserved
0x7	1 Msps
0x8 - 0xF	Reserved

Register 56: ADC Peripheral Configuration (ADCPC), offset 0xFC4

The **ADCPC** register provides information regarding the configuration of the peripheral.

ADC Peripheral Configuration (ADCPC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xFC4
Type RW, reset 0x0000.0007

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												SR			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	SR	RW	0x7	ADC Sample Rate This field specifies the number of ADC conversions per second and is used in Run, Sleep, and Deep-Sleep modes. The field encoding is based on the legacy **RCGC0** register encoding. The programmed sample rate cannot exceed the maximum sample rate specified by the `MSR` field in the **ADCPP** register. The `SR` field is encoded as follows:

Value	Description
0x0	Reserved
0x1	125 ksps
0x2	Reserved
0x3	250 ksps
0x4	Reserved
0x5	500 ksps
0x6	Reserved
0x7	1 Msps
0x8 - 0xF	Reserved

Register 57: ADC Clock Configuration (ADCCC), offset 0xFC8

The **ADCCC** register controls the clock source for the ADC module.

To use the PIOSC to clock the ADC, first power up the PLL and then enable the PIOSC in the CS bit field, then disable the PLL.

To use the MOSC to clock the ADC, first power up the PLL and then enable the clock to the ADC module, then disable the PLL and switch to the MOSC for the system clock.

ADC Clock Configuration (ADCCC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xFC8
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												CS			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	CS	RW	0	ADC Clock Source The following table specifies the clock source that generates the ADC clock input, see Figure 5-5 on page 210.

Value	Description
0x0	Either the 16-MHz system clock (if the PLL bypass is in effect) or the 16 MHz clock derived from PLL ÷ 25 (default). Note that when the PLL is bypassed, the system clock must be at least 16 MHz.
0x1	PIOSC The PIOSC provides a 16-MHz clock source for the ADC. If the PIOSC is used as the clock source, the ADC module can continue to operate in Deep-Sleep mode.
0x2 - 0xF	Reserved

13 Universal Asynchronous Receivers/Transmitters (UARTs)

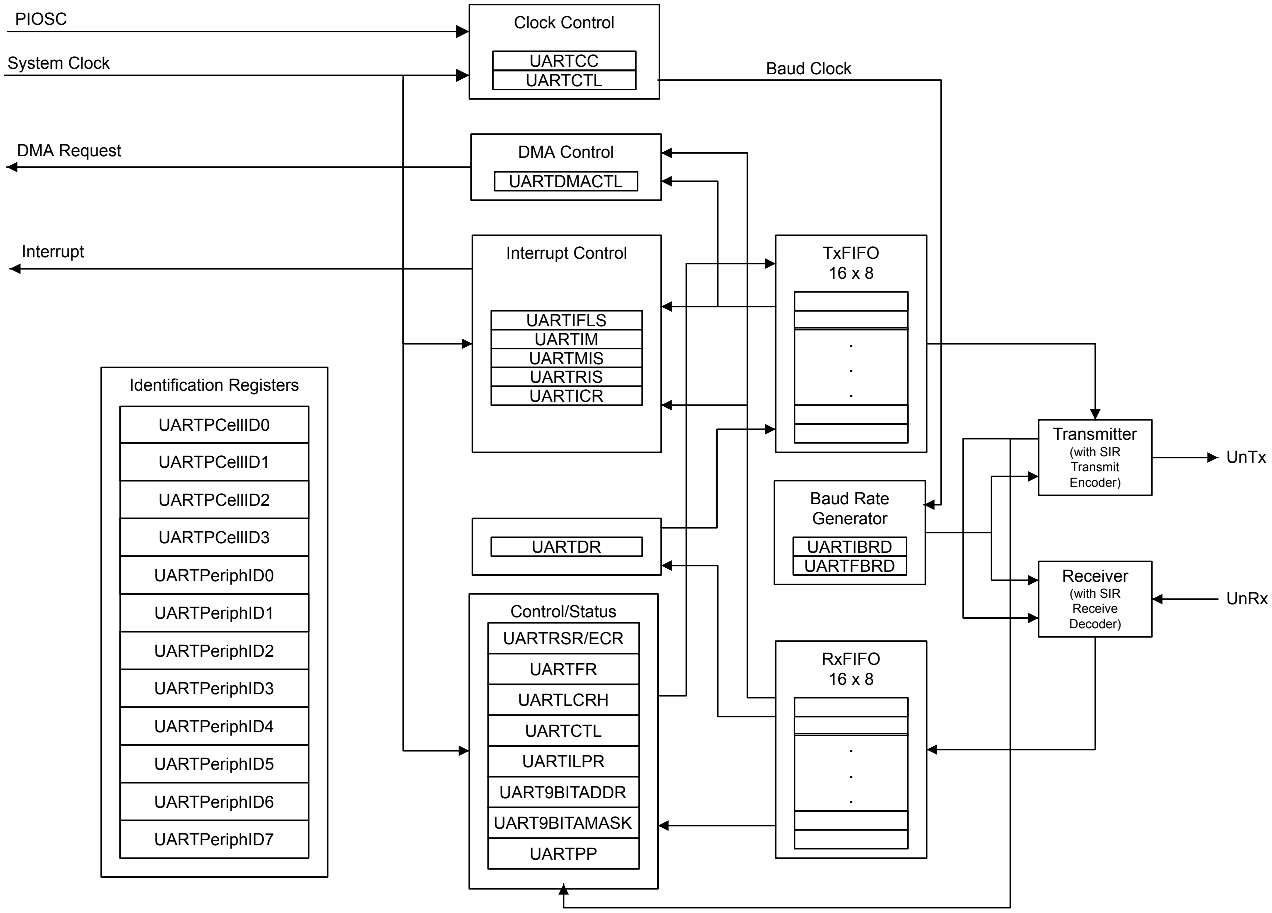
The TM4C1232C3PM controller includes eight Universal Asynchronous Receiver/Transmitter (UART) with the following features:

- Programmable baud-rate generator allowing speeds up to 5 Mbps for regular speed (divide by 16) and 10 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- Line-break generation and detection
- Fully programmable serial interface characteristics
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no-parity bit generation/detection
  - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder/decoder providing
  - Programmable use of IrDA Serial Infrared (SIR) or UART input/output
  - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
  - Support of normal 3/16 and low-power (1.41-2.23 μs) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- Modem flow control (on UART1)
- EIA-485 9-bit support
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (μDMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level

– Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

13.1 Block Diagram

Figure 13-1. UART Module Block Diagram



13.2 Signal Description

The following table lists the external signals of the UART module and describes the function of each. The UART signals are alternate functions for some GPIO signals and default to be GPIO signals at reset, with the exception of the U0Rx and U0Tx pins which default to the UART function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these UART signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 603) should be set to choose the UART function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 620) 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 581.

Table 13-1. UART Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
U0Rx	17	PA0 (1)	I	TTL	UART module 0 receive.
U0Tx	18	PA1 (1)	O	TTL	UART module 0 transmit.
U1CTS	15 29	PC5 (8) PF1 (1)	I	TTL	UART module 1 Clear To Send modem flow control input signal.
U1RTS	16 28	PC4 (8) PF0 (1)	O	TTL	UART module 1 Request to Send modem flow control output line.
U1Rx	16 45	PC4 (2) PB0 (1)	I	TTL	UART module 1 receive.
U1Tx	15 46	PC5 (2) PB1 (1)	O	TTL	UART module 1 transmit.
U2Rx	33 53	PG4 (1) PD6 (1)	I	TTL	UART module 2 receive.
U2Tx	10 32	PD7 (1) PG5 (1)	O	TTL	UART module 2 transmit.
U3Rx	14	PC6 (1)	I	TTL	UART module 3 receive.
U3Tx	13	PC7 (1)	O	TTL	UART module 3 transmit.
U4Rx	16	PC4 (1)	I	TTL	UART module 4 receive.
U4Tx	15	PC5 (1)	O	TTL	UART module 4 transmit.
U5Rx	59	PE4 (1)	I	TTL	UART module 5 receive.
U5Tx	60	PE5 (1)	O	TTL	UART module 5 transmit.
U6Rx	43	PD4 (1)	I	TTL	UART module 6 receive.
U6Tx	44	PD5 (1)	O	TTL	UART module 6 transmit.
U7Rx	9	PE0 (1)	I	TTL	UART module 7 receive.
U7Tx	8	PE1 (1)	O	TTL	UART module 7 transmit.

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

13.3 Functional Description

Each TM4C1232C3PM 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 847). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the UARTEN bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

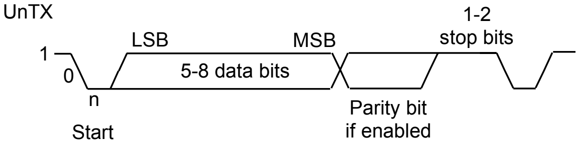
The UART module also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

13.3.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit and followed by the data bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 13-2 on page 825 for details.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO.

Figure 13-2. UART Character Frame



13.3.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divisor allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 843) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 844). The baud-rate divisor (BRD) has the following relationship to the system clock (where *BRDI* is the integer part of the BRD and *BRDF* is the fractional part, separated by a decimal place.)

```
BRD = BRDI + BRDF = UARTSysClk / (ClkDiv * Baud Rate)
```

where UARTSysClk is the system clock connected to the UART, and ClkDiv is either 16 (if HSE in **UARTCTL** is clear) or 8 (if HSE is set). By default, this will be the main system clock described in “Clock Control” on page 208. Alternatively, the UART may be clocked from the internal precision oscillator (PIOSC), independent of the system clock selection. This will allow the UART clock to be programmed independently of the system clock PLL settings. See the **UARTCC** register for more details.

The 6-bit fractional number (that is to be loaded into the DIVFRAC bit field in the **UARTFBRD** register) can be calculated by taking the fractional part of the baud-rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors:

```
UARTFBRD[DIVFRAC] = integer(BRDF * 64 + 0.5)
```

The UART generates an internal baud-rate reference clock at 8x or 16x the baud-rate (referred to as Baud8 and Baud16, depending on the setting of the HSE bit (bit 5) in **UARTCTL**). This reference clock is divided by 8 or 16 to generate the transmit clock, and is used for error detection during receive operations. Note that the state of the HSE bit has no effect on clock generation in ISO 7816 smart card mode (when the SMART bit in the **UARTCTL** register is set).

Along with the **UART Line Control, High Byte (UARTLCRH)** register (see page 845), the **UARTIBRD** and **UARTFBRD** registers form an internal 30-bit register. This internal register is only updated when a write operation to **UARTLCRH** is performed, so any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register for the changes to take effect.

To update the baud-rate registers, there are four possible sequences:

- **UARTIBRD** write, **UARTFBRD** write, and **UARTLCRH** write
- **UARTFBRD** write, **UARTIBRD** write, and **UARTLCRH** write
- **UARTIBRD** write and **UARTLCRH** write

- **UARTFBRD** write and **UARTLCRH** write

13.3.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit FIFO. The BUSY bit in the **UART Flag (UARTFR)** register (see page 840) 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 824).

The start bit is valid and recognized if the UnRx signal is still low on the eighth cycle of Baud16 (HSE clear) or the fourth cycle of Baud 8 (HSE set), otherwise it is ignored. After a valid start bit is detected, successive data bits are sampled on every 16th cycle of Baud16 or 8th cycle of Baud8 (that is, one bit period later) according to the programmed length of the data characters and value of the HSE bit in **UARTCTL**. The parity bit is then checked if parity mode is enabled. Data length and parity are defined in the **UARTLCRH** register.

Lastly, a valid stop bit is confirmed if the UnRx signal is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO along with any error bits associated with that word.

13.3.4 Serial IR (SIR)

The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream and a half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output and decoded input to the UART. When enabled, the SIR block uses the UnTx and UnRx pins for the SIR protocol. These signals should be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as a high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW and driving the UART input pin LOW.

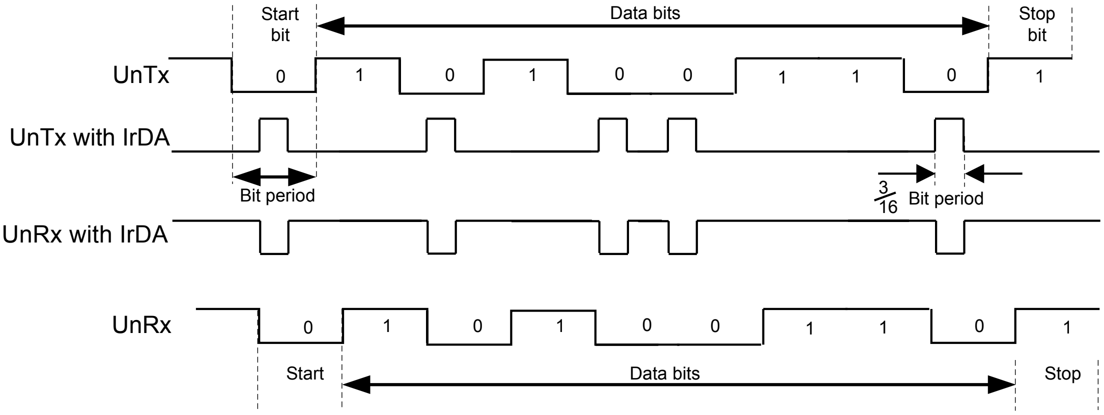
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated IrLPBaud16 signal (1.63 µs, assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the **UARTCTL** register (see page 847).

Whether the device is in normal or low-power IrDA mode, a start bit is deemed valid if the decoder is still Low, one period of IrLPBaud16 after the Low was first detected. This enables a normal-mode UART to receive data from a low-power mode UART that can transmit pulses as small as 1.41 µs.

Thus, for both low-power and normal mode operation, the ILPDVSR field in the **UARTILPR** register must be programmed such that 1.42 MHz < $F_{\mathtt{IrLPBaud16}}$ < 2.12 MHz, resulting in a low-power pulse duration of 1.41–2.11 μs (three times the period of IrLPBaud16). The minimum frequency of IrLPBaud16 ensures that pulses less than one period of IrLPBaud16 are rejected, but pulses greater than 1.4 μs are accepted as valid pulses.

Figure 13-3 on page 827 shows the UART transmit and receive signals, with and without IrDA modulation.

Figure 13-3. IrDA Data Modulation



In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10-ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency or receiver setup time.

13.3.5 ISO 7816 Support

The UART offers basic support to allow communication with an ISO 7816 smartcard. When bit 3 (SMART) of the **UARTCTL** register is set, the UnTx signal is used as a bit clock, and the UnRx signal is used as the half-duplex communication line connected to the smartcard. A GPIO signal can be used to generate the reset signal to the smartcard. The remaining smartcard signals should be provided by the system design. The maximum clock rate in this mode is system clock / 16.

When using ISO 7816 mode, the **UARTLCRH** register must be set to transmit 8-bit words (WLEN bits 6:5 configured to 0x3) with EVEN parity (PEN set and EPS set). In this mode, the UART automatically uses 2 stop bits, and the STP2 bit of the **UARTLCRH** register is ignored.

If a parity error is detected during transmission, UnRx is pulled Low during the second stop bit. In this case, the UART aborts the transmission, flushes the transmit FIFO and discards any data it contains, and raises a parity error interrupt, allowing software to detect the problem and initiate retransmission of the affected data. Note that the UART does not support automatic retransmission in this case.

13.3.6 Modem Handshake Support

This section describes how to configure and use the modem flow control signals for UART1 when connected as a DTE (data terminal equipment) or as a DCE (data communications equipment). In general, a modem is a DCE and a computing device that connects to a modem is the DTE.

13.3.6.1 Signaling

The status signals provided by UART1 differ based on whether the UART is used as a DTE or DCE. When used as a DTE, the modem flow control signals are defined as:

- $\overline{\mathtt{U1CTS}}$ is Clear To Send
- $\overline{\mathtt{U1RTS}}$ is Request To Send

When used as a DCE, the modem flow control signals are defined as:

- $\overline{\mathtt{U1CTS}}$ is Request To Send
- $\overline{\mathtt{U1RTS}}$ is Clear To Send

13.3.6.2 Flow Control

Flow control can be accomplished by either hardware or software. The following sections describe the different methods.

Hardware Flow Control (RTS/CTS)

Hardware flow control between two devices is accomplished by connecting the $\overline{\mathtt{U1RTS}}$ output to the Clear-To-Send input on the receiving device, and connecting the Request-To-Send output on the receiving device to the $\overline{\mathtt{U1CTS}}$ input.

The $\overline{\mathtt{U1CTS}}$ input controls the transmitter. The transmitter may only transmit data when the $\overline{\mathtt{U1CTS}}$ input is asserted. The $\overline{\mathtt{U1RTS}}$ output signal indicates the state of the receive FIFO. $\overline{\mathtt{U1CTS}}$ remains asserted until the preprogrammed watermark level is reached, indicating that the Receive FIFO has no space to store additional characters.

The **UARTCTL** register bits 15 (CTSEN) and 14 (RTSEN) specify the flow control mode as shown in Table 13-2 on page 828.

Table 13-2. 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 $\overline{\mathtt{U1RTS}}$ output value through the **UARTCTL** register Request to Send (RTS) bit, and the status of the RTS bit should be ignored.

Software Flow Control (Modem Status Interrupts)

Software flow control between two devices is accomplished by using interrupts to indicate the status of the UART. Interrupts may be generated for the $\overline{\mathtt{U1CTS}}$ signal using bit 3 of the **UARTIM** register. The raw and masked interrupt status may be checked using the **UARTRIS** and **UARTMIS** register. These interrupts may be cleared using the **UARTICR** register.

13.3.7 9-Bit UART Mode

The UART provides a 9-bit mode that is enabled with the 9BITEN bit in the **UART9BITADDR** register. This feature is useful in a multi-drop configuration of the UART where a single master connected to multiple slaves can communicate with a particular slave through its address or set of addresses along with a qualifier for an address byte. All the slaves check for the address qualifier in the place of the parity bit and, if set, then compare the byte received with the preprogrammed address. If the address matches, then it receives or sends further data. If the address does not match, it drops the address byte and any subsequent data bytes. If the UART is in 9-bit mode, then the receiver operates with no parity mode. The address can be predefined to match with the received byte and it can be configured with the **UART9BITADDR** register. The matching can be extended to a set of addresses using the address mask in the **UART9BITAMASK** register. By default, the **UART9BITAMASK** is 0xFF, meaning that only the specified address is matched.

When not finding a match, the rest of the data bytes with the 9th bit cleared are dropped. If a match is found, then an interrupt is generated to the NVIC for further action. The subsequent data bytes with the cleared 9th bit are stored in the FIFO. Software can mask this interrupt in case μDMA and/or FIFO operations are enabled for this instance and processor intervention is not required. All the send transactions with 9-bit mode are data bytes and the 9th bit is cleared. Software can override the 9th bit to be set (to indicate address) by overriding the parity settings to sticky parity with odd parity enabled for a particular byte. To match the transmission time with correct parity settings, the address byte can be transmitted as a single then a burst transfer. The Transmit FIFO does not hold the address/data bit, hence software should take care of enabling the address bit appropriately.

13.3.8 FIFO Operation

The UART has two 16x8 FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the **UART Data (UARTDR)** register (see page 835). 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 845).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 840) and the **UART Receive Status (UARTRSR)** register. Hardware monitors empty, full and overrun conditions. The **UARTFR** register contains empty and full flags (TXFE, TXFF, RXFE, and RXFF bits), and the **UARTRSR** register shows overrun status via the OE bit. If the FIFOs are disabled, the empty and full flags are set according to the status of the 1-byte-deep holding registers.

The trigger points at which the FIFOs generate interrupts is controlled via the **UART Interrupt FIFO Level Select (UARTIFLS)** register (see page 851). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include ⅛, ¼, ½, ¾, and ⅞. For example, if the ¼ option is selected for the receive FIFO, the UART generates a receive interrupt after 4 data bytes are received. Out of reset, both FIFOs are configured to trigger an interrupt at the ½ mark.

13.3.9 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun Error
- Break Error
- Parity Error

- Framing Error
- Receive Timeout
- Transmit (when condition defined in the TXIFLSEL bit in the **UARTIFLS** register is met, or if the EOT bit in **UARTCTL** is set, when the last bit of all transmitted data leaves the serializer)
- Receive (when condition defined in the RXIFLSEL bit in the **UARTIFLS** register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 859).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 853) by setting the corresponding IM bits. If interrupts are not used, the raw interrupt status is visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 856).

Note: For receive timeout, the RTIM bit in the **UARTIM** register must be set to see the RTMIS and RTRIS status in the **UARTMIS** and **UARTRIS** registers.

Interrupts are always cleared (for both the **UARTMIS** and **UARTRIS** registers) by writing a 1 to the corresponding bit in the **UART Interrupt Clear (UARTICR)** register (see page 862).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period when the HSE bit is clear or over a 64-bit period when the HSE bit is set. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the **UARTICR** register.

The receive interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the receive FIFO reaches the programmed trigger level, the RXRIS bit is set. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, or by clearing the interrupt by writing a 1 to the RXIC bit.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the RXRIS bit is set. The receive interrupt is cleared by performing a single read of the receive FIFO, or by clearing the interrupt by writing a 1 to the RXIC bit.

The transmit interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO progresses through the programmed trigger level, the TXRIS bit is set. The transmit interrupt is based on a transition through level, therefore the FIFO must be written past the programmed trigger level otherwise no further transmit interrupts will be generated. The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, or by clearing the interrupt by writing a 1 to the TXIC bit.
- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the TXRIS bit is set. It is cleared by performing a single write to the transmit FIFO, or by clearing the interrupt by writing a 1 to the TXIC bit.

13.3.10 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work by setting the LBE bit in the **UARTCTL** register (see page 847). In loopback mode, data transmitted on the

UnTx output is received on the UnRx input. Note that the LBE bit should be set before the UART is enabled.

13.3.11 DMA Operation

The UART provides an interface to the µDMA controller with separate channels for transmit and receive. The DMA operation of the UART is enabled through the **UART DMA Control (UARTDMACTL)** register. When DMA operation is enabled, the UART asserts a DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever any data is in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is at or above the FIFO trigger level configured in the **UARTIFLS** register. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO contains fewer characters than the FIFO trigger level. The single and burst DMA transfer requests are handled automatically by the µDMA controller depending on how the DMA channel is configured.

To enable DMA operation for the receive channel, set the RXDMAE bit of the **DMA Control (UARTDMACTL)** register. To enable DMA operation for the transmit channel, set the TXDMAE bit of the **UARTDMACTL** register. The UART can also be configured to stop using DMA for the receive channel if a receive error occurs. If the DMAERR bit of the **UARTDMACR** register is set and a receive error occurs, the DMA receive requests are automatically disabled. This error condition can be cleared by clearing the appropriate UART error interrupt.

If the µDMA is enabled, then the controller triggers an interrupt when the TX FIFO or RX FIFO has reached a trigger point as programmed in the **UARTIFLS** register. The interrupt occurs on the UART interrupt vector. Therefore, if interrupts are used for UART operation and DMA is enabled, the UART interrupt handler must be designed to handle the µDMA completion interrupt.

Note: To trigger an interrupt on transmit completion from the UART's serializer, the EOT bit must be set in the **UARTCTL** register. In this configuration, the transmit interrupt is generated once the FIFO is completely empty and all data including the stop bits have left the transmit serializer. In this case, setting the TXIFLSEL bit in the **UARTIFLS** register is ignored.

When transfers are performed from a FIFO of the UART using the µDMA, and any interrupt is generated from the UART, the UART module's status bit in the **DMA Channel Interrupt Status (DMACHIS)** register must be checked at the end of the interrupt service routine. If the status bit is set, clear the interrupt by writing a 1 to it.

See “Micro Direct Memory Access (µDMA)” on page 517 for more details about programming the µDMA controller.

13.4 Initialization and Configuration

To enable and initialize the UART, the following steps are necessary:

1. Enable the UART module using the **RCGCUART** register (see page 322).

2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register (see page 319). To find out which GPIO port to enable, refer to Table 20-5 on page 1115.

3. Set the GPIO AFSEL bits for the appropriate pins (see page 603). To determine which GPIOs to configure, see Table 20-4 on page 1111.

4. Configure the GPIO current level and/or slew rate as specified for the mode selected (see page 605 and page 613).

5. Configure the PMCn fields in the **GPIOPCTL** register to assign the UART signals to the appropriate pins (see page 620 and Table 20-5 on page 1115).

To use the UART, the peripheral clock must be enabled by setting the appropriate bit in the **RCGCUART** register (page 322). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGCGPIO** register (page 319) in the System Control module. To find out which GPIO port to enable, refer to Table 20-5 on page 1115.

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 825, the BRD can be calculated:

```
BRD = 20,000,000 / (16 * 115,200) = 10.8507
```

which means that the DIVINT field of the **UARTIBRD** register (see page 843) should be set to 10 decimal or 0xA. The value to be loaded into the **UARTFBRD** register (see page 844) is calculated by the equation:

```
UARTFBRD[DIVFRAC] = integer(0.8507 * 64 + 0.5) = 54
```

With the BRD values in hand, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the UARTEN bit in the **UARTCTL** register.
2. Write the integer portion of the BRD to the **UARTIBRD** register.
3. Write the fractional portion of the BRD to the **UARTFBRD** register.
4. Write the desired serial parameters to the **UARTLCRH** register (in this case, a value of 0x0000.0060).
5. Configure the UART clock source by writing to the **UARTCC** register.
6. Optionally, configure the µDMA channel (see “Micro Direct Memory Access (µDMA)” on page 517) and enable the DMA option(s) in the **UARTDMACTL** register.
7. Enable the UART by setting the UARTEN bit in the **UARTCTL** register.

13.5 Register Map

Table 13-3 on page 833 lists the UART registers. The offset listed is a hexadecimal increment to the register's address, relative to that UART's base address:

- UART0: 0x4000.C000
- UART1: 0x4000.D000
- UART2: 0x4000.E000
- UART3: 0x4000.F000
- UART4: 0x4001.0000
- UART5: 0x4001.1000
- UART6: 0x4001.2000
- UART7: 0x4001.3000

The UART module clock must be enabled before the registers can be programmed (see page 322). There must be a delay of 3 system clocks after the UART module clock is enabled before any UART module registers are accessed.

The UART must be disabled (see the UARTEN bit in the **UARTCTL** register on page 847) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

Table 13-3. UART Register Map

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	RW	0x0000.0000	UART Data	835
0x004	UARTRSR/UARTECR	RW	0x0000.0000	UART Receive Status/Error Clear	837
0x018	UARTFR	RO	0x0000.0090	UART Flag	840
0x020	UARTILPR	RW	0x0000.0000	UART IrDA Low-Power Register	842
0x024	UARTIBRD	RW	0x0000.0000	UART Integer Baud-Rate Divisor	843
0x028	UARTFBRD	RW	0x0000.0000	UART Fractional Baud-Rate Divisor	844
0x02C	UARTLCRH	RW	0x0000.0000	UART Line Control	845
0x030	UARTCTL	RW	0x0000.0300	UART Control	847
0x034	UARTIFLS	RW	0x0000.0012	UART Interrupt FIFO Level Select	851
0x038	UARTIM	RW	0x0000.0000	UART Interrupt Mask	853
0x03C	UARTRIS	RO	0x0000.0000	UART Raw Interrupt Status	856
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	859
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	862
0x048	UARTDMACTL	RW	0x0000.0000	UART DMA Control	864
0x0A4	UART9BITADDR	RW	0x0000.0000	UART 9-Bit Self Address	865
0x0A8	UART9BITAMASK	RW	0x0000.00FF	UART 9-Bit Self Address Mask	866
0xFC0	UARTPP	RO	0x0000.0003	UART Peripheral Properties	867
0xFC8	UARTCC	RW	0x0000.0000	UART Clock Configuration	868
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	869
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	870
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	871

Table 13-3. UART Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	872
0xFE0	UARTPeriphID0	RO	0x0000.0060	UART Peripheral Identification 0	873
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	874
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	875
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	876
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	877
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	878
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	879
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	880

13.6 Register Descriptions

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

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

Important: This register is read-sensitive. See the register description for details.

This register is the data register (the interface to the FIFOs).

For transmitted data, if the FIFO is enabled, data written to this location is pushed onto the transmit FIFO. If the FIFO is disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. If the FIFO is disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

UART Data (UARTDR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x000
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				OE	BE	PE	FE	DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	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
0	No data has been lost due to a FIFO overrun.
1	New data was received when the FIFO was full, resulting in data loss.

Bit/Field	Name	Type	Reset	Description
10	BE	RO	0	UART Break Error Value Description 0 No break condition has occurred 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). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the received data input goes to a 1 (marking state), and the next valid start bit is received.
9	PE	RO	0	UART Parity Error Value Description 0 No parity error has occurred 1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. In FIFO mode, this error is associated with the character at the top of the FIFO.
8	FE	RO	0	UART Framing Error Value Description 0 No framing error has occurred 1 The received character does not have a valid stop bit (a valid stop bit is 1).
7:0	DATA	RW	0x00	Data Transmitted or Received Data that is to be transmitted via the UART is written to this field. When read, this field contains the data that was received by the UART.

Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004

The **UARTRSR/UARTECR** register is the receive status register/error clear register.

In addition to the **UARTDR** register, receive status can also be read from the **UARTRSR** register. If the status is read from this register, then the status information corresponds to the entry read from **UARTDR** prior to reading **UARTRSR**. The status information for overrun is set immediately when an overrun condition occurs.

The **UARTRSR** register cannot be written.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared on reset.

Read-Only Status Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x004
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												OE	BE	PE	FE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	OE	RO	0	UART Overrun Error Value Description 0 No data has been lost due to a FIFO overrun. 1 New data was received when the FIFO was full, resulting in data loss. 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 0 No break condition has occurred 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). 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 0 No parity error has occurred 1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. This bit is cleared to 0 by a write to **UARTECR**.
0	FE	RO	0	UART Framing Error Value Description 0 No framing error has occurred 1 The received character does not have a valid stop bit (a valid stop bit is 1). This bit is cleared to 0 by a write to **UARTECR**. In FIFO mode, this error is associated with the character at the top of the FIFO.

Write-Only Error Clear Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x004
Type WO, reset 0x0000.0000

	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 CTS bit indicate the modem flow control. Note that the modem bits are only implemented on UART1 and are reserved on UART0 and UART2.

UART Flag (UARTFR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x018
Type RO, reset 0x0000.0090

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								TXFE	RXFF	TXFF	RXFE	BUSY	reserved		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
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TXFE	RO	1	UART Transmit FIFO Empty The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. Value Description 0 The transmitter has data to transmit. 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.
6	RXFF	RO	0	UART Receive FIFO Full The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. Value Description 0 The receiver can receive data. 1 If the FIFO is disabled (FEN is 0), the receive holding register is full. If the FIFO is enabled (FEN is 1), the receive FIFO is full.

Bit/Field	Name	Type	Reset	Description
5	TXFF	RO	0	UART Transmit FIFO Full The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. Value Description 0 The transmitter is not full. 1 If the FIFO is disabled (FEN is 0), the transmit holding register is full. If the FIFO is enabled (FEN is 1), the transmit FIFO is full.
4	RXFE	RO	1	UART Receive FIFO Empty The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. Value Description 0 The receiver is not empty. 1 If the FIFO is disabled (FEN is 0), the receive holding register is empty. If the FIFO is enabled (FEN is 1), the receive FIFO is empty.
3	BUSY	RO	0	UART Busy Value Description 0 The UART is not busy. 1 The UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).
2:1	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	CTS	RO	0	Clear To Send Value Description 0 The U1CTS signal is not asserted. 1 The U1CTS signal is asserted.

Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register stores the 8-bit low-power counter divisor value used to derive the low-power SIR pulse width clock by dividing down the system clock (SysClk). All the bits are cleared when reset.

The internal `IrLPBaud16` clock is generated by dividing down SysClk according to the low-power divisor value written to **UARTILPR**. The duration of SIR pulses generated when low-power mode is enabled is three times the period of the `IrLPBaud16` clock. The low-power divisor value is calculated as follows:

$\mathtt{ILPDVSR = SysClk / F_{IrLPBaud16}}$

where $F_{\mathtt{IrLPBaud16}}$ is nominally 1.8432 MHz.

Because the `IrLPBaud16` clock is used to sample transmitted data irrespective of mode, the `ILPDVSR` field must be programmed in both low power and normal mode,such that 1.42 MHz < $F_{\mathtt{IrLPBaud16}}$ < 2.12 MHz, resulting in a low-power pulse duration of 1.41–2.11 µs (three times the period of `IrLPBaud16`). The minimum frequency of `IrLPBaud16` ensures that pulses less than one period of `IrLPBaud16` are rejected, but pulses greater than 1.4 µs are accepted as valid pulses.

Note: Zero is an illegal value. Programming a zero value results in no `IrLPBaud16` pulses being generated.

UART IrDA Low-Power Register (UARTILPR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x020
Type RW, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ILPDVSR	RW	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 825 for configuration details.

UART Integer Baud-Rate Divisor (UARTIBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x024
Type RW, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DIVINT	RW	0x0000	Integer Baud-Rate Divisor

Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028

The **UARTFBRD** register is the fractional part of the baud-rate divisor value. All the bits are cleared on reset. When changing the **UARTFBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 825 for configuration details.

UART Fractional Baud-Rate Divisor (UARTFBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x028
Type RW, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	DIVFRAC	RW	0x0	Fractional Baud-Rate Divisor

Register 7: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x02C
Type RW, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	SPS	RW	0	UART Stick Parity Select When bits 1, 2, and 7 of **UARTLCRH** are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set and 2 is cleared, the parity bit is transmitted and checked as a 1. When this bit is cleared, stick parity is disabled.
6:5	WLEN	RW	0x0	UART Word Length The bits indicate the number of data bits transmitted or received in a frame as follows:

Value	Description
0x0	5 bits (default)
0x1	6 bits
0x2	7 bits
0x3	8 bits

Bit/Field	Name	Type	Reset	Description
4	FEN	RW	0	UART Enable FIFOs Value Description 0 The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers. 1 The transmit and receive FIFO buffers are enabled (FIFO mode).
3	STP2	RW	0	UART Two Stop Bits Select Value Description 0 One stop bit is transmitted at the end of a frame. 1 Two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received. When in 7816 smartcard mode (the SMART bit is set in the **UARTCTL** register), the number of stop bits is forced to 2.
2	EPS	RW	0	UART Even Parity Select Value Description 0 Odd parity is performed, which checks for an odd number of 1s. 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. This bit has no effect when parity is disabled by the PEN bit.
1	PEN	RW	0	UART Parity Enable Value Description 0 Parity is disabled and no parity bit is added to the data frame. 1 Parity checking and generation is enabled.
0	BRK	RW	0	UART Send Break Value Description 0 Normal use. 1 A Low level is continually output on the UnTx signal, after completing transmission of the current character. For the proper execution of the break command, software must set this bit for at least two frames (character periods).

Register 8: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the Transmit Enable (TXE) and Receive Enable (RXE) bits, which are set.

To enable the UART module, the UARTEN bit must be set. If software requires a configuration change in the module, the UARTEN bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

Note: The **UARTCTL** register should not be changed while the UART is enabled or else the results are unpredictable. The following sequence is recommended for making changes to the **UARTCTL** register.

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by clearing bit 4 (FEN) in the line control register (**UARTLCRH**).
4. Reprogram the control register.
5. Enable the UART.

UART Control (UARTCTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x030
Type RW, reset 0x0000.0300

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CTSEN	RTSEN	reserved		RTS	reserved	RXE	TXE	LBE	reserved	HSE	EOT	SMART	SIRLP	SIREN	UARTEN
Type	RW	RW	RO	RO	RW	RO	RW	RW	RW	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	CTSEN	RW	0	Enable Clear To Send

Value	Description
0	CTS hardware flow control is disabled.
1	CTS hardware flow control is enabled. Data is only transmitted when the U1CTS signal is asserted.

Bit/Field	Name	Type	Reset	Description
14	RTSEN	RW	0	Enable Request to Send Value Description 0 RTS hardware flow control is disabled. 1 RTS hardware flow control is enabled. Data is only requested (by asserting U1RTS) when the receive FIFO has available entries.
13:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	RTS	RW	0	Request to Send When RTSEN is clear, the status of this bit is reflected on the U1RTS signal. If RTSEN is set, this bit is ignored on a write and should be ignored on read.
10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	RXE	RW	1	UART Receive Enable Value Description 0 The receive section of the UART is disabled. 1 The receive section of the UART is enabled. If the UART is disabled in the middle of a receive, it completes the current character before stopping. **Note:** To enable reception, the UARTEN bit must also be set.
8	TXE	RW	1	UART Transmit Enable Value Description 0 The transmit section of the UART is disabled. 1 The transmit section of the UART is enabled. If the UART is disabled in the middle of a transmission, it completes the current character before stopping. **Note:** To enable transmission, the UARTEN bit must also be set.
7	LBE	RW	0	UART Loop Back Enable Value Description 0 Normal operation. 1 The UnTx path is fed through the UnRx path.
6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
5	HSE	RW	0	High-Speed Enable Value Description 0 The UART is clocked using the system clock divided by 16. 1 The UART is clocked using the system clock divided by 8. **Note:** System clock used is also dependent on the baud-rate divisor configuration (see page 843) and page 844). The state of this bit has no effect on clock generation in ISO 7816 smart card mode (the `SMART` bit is set).
4	EOT	RW	0	End of Transmission This bit determines the behavior of the `TXRIS` bit in the **UARTRIS** register. Value Description 0 The `TXRIS` bit is set when the transmit FIFO condition specified in **UARTIFLS** is met. 1 The `TXRIS` bit is set only after all transmitted data, including stop bits, have cleared the serializer.
3	SMART	RW	0	ISO 7816 Smart Card Support Value Description 0 Normal operation. 1 The UART operates in Smart Card mode. The application must ensure that it sets 8-bit word length (`WLEN` set to 0x3) and even parity (`PEN` set to 1, `EPS` set to 1, `SPS` set to 0) in **UARTLCRH** when using ISO 7816 mode. In this mode, the value of the `STP2` bit in **UARTLCRH** is ignored and the number of stop bits is forced to 2. Note that the UART does not support automatic retransmission on parity errors. If a parity error is detected on transmission, all further transmit operations are aborted and software must handle retransmission of the affected byte or message.
2	SIRLP	RW	0	UART SIR Low-Power Mode This bit selects the IrDA encoding mode. Value Description 0 Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period. 1 The UART operates in SIR Low-Power mode. Low-level bits are transmitted with a pulse width which is 3 times the period of the `IrLPBaud16` input signal, regardless of the selected bit rate. Setting this bit uses less power, but might reduce transmission distances. See page 842 for more information.

Bit/Field	Name	Type	Reset	Description
1	SIREN	RW	0	UART SIR Enable Value Description 0 Normal operation. 1 The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.
0	UARTEN	RW	0	UART Enable Value Description 0 The UART is disabled. 1 The UART is enabled. If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034

The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the TXRIS and RXRIS bits in the **UARTRIS** register are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

Out of reset, the TXIFLSEL and RXIFLSEL bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x034
Type RW, reset 0x0000.0012

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	RW	RW	RW	RW	RW	RW
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	RW	0x2	UART Receive Interrupt FIFO Level Select The trigger points for the receive interrupt are as follows:

Value	Description
0x0	RX FIFO ≥ ⅛ full
0x1	RX FIFO ≥ ¼ full
0x2	RX FIFO ≥ ½ full (default)
0x3	RX FIFO ≥ ¾ full
0x4	RX FIFO ≥ ⅞ full
0x5-0x7	Reserved

Bit/Field	Name	Type	Reset	Description
2:0	TXIFLSEL	RW	0x2	UART Transmit Interrupt FIFO Level Select The trigger points for the transmit interrupt are as follows: Value Description 0x0 TX FIFO ≤ ⅞ empty 0x1 TX FIFO ≤ ¾ empty 0x2 TX FIFO ≤ ½ empty (default) 0x3 TX FIFO ≤ ¼ empty 0x4 TX FIFO ≤ ⅛ empty 0x5-0x7 Reserved **Note:** If the `EOT` bit in **UARTCTL** is set (see page 847), 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.

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x038
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			9BITIM	reserved	OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	reserved		CTSIM	reserved
Type	RO	RO	RO	RW	RO	RW	RW	RW	RW	RW	RW	RW	RO	RO	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	9BITIM	RW	0	9-Bit Mode Interrupt Mask Value Description 0 The 9BITRIS interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the 9BITRIS bit in the **UARTRIS** register is set.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIM	RW	0	UART Overrun Error Interrupt Mask Value Description 0 The OERIS interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the OERIS bit in the **UARTRIS** register is set.

Bit/Field	Name	Type	Reset	Description
9	BEIM	RW	0	UART Break Error Interrupt Mask Value Description 0 The `BERIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `BERIS` bit in the **UARTRIS** register is set.
8	PEIM	RW	0	UART Parity Error Interrupt Mask Value Description 0 The `PERIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `PERIS` bit in the **UARTRIS** register is set.
7	FEIM	RW	0	UART Framing Error Interrupt Mask Value Description 0 The `FERIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `FERIS` bit in the **UARTRIS** register is set.
6	RTIM	RW	0	UART Receive Time-Out Interrupt Mask Value Description 0 The `RTRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `RTRIS` bit in the **UARTRIS** register is set.
5	TXIM	RW	0	UART Transmit Interrupt Mask Value Description 0 The `TXRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `TXRIS` bit in the **UARTRIS** register is set.
4	RXIM	RW	0	UART Receive Interrupt Mask Value Description 0 The `RXRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `RXRIS` bit in the **UARTRIS** register is set.

Bit/Field	Name	Type	Reset	Description
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	CTSIM	RW	0	UART Clear to Send Modem Interrupt Mask Value Description 0 The `CTSRIS` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `CTSRIS` bit in the **UARTRIS** register is set. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x03C
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			9BITRIS	reserved	OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	reserved		CTSRIS	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:13	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	9BITRIS	RO	0	9-Bit Mode Raw Interrupt Status Value Description 0 No interrupt 1 A receive address match has occurred. This bit is cleared by writing a 1 to the `9BITIC` bit in the **UARTICR** register.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OERIS	RO	0	UART Overrun Error Raw Interrupt Status Value Description 0 No interrupt 1 An overrun error has occurred. This bit is cleared by writing a 1 to the `OEIC` bit in the **UARTICR** register.
9	BERIS	RO	0	UART Break Error Raw Interrupt Status Value Description 0 No interrupt 1 A break error has occurred. This bit is cleared by writing a 1 to the `BEIC` bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
8	PERIS	RO	0	UART Parity Error Raw Interrupt Status Value Description 0 No interrupt 1 A parity error has occurred. This bit is cleared by writing a 1 to the PEIC bit in the **UARTICR** register.
7	FERIS	RO	0	UART Framing Error Raw Interrupt Status Value Description 0 No interrupt 1 A framing error has occurred. This bit is cleared by writing a 1 to the FEIC bit in the **UARTICR** register.
6	RTRIS	RO	0	UART Receive Time-Out Raw Interrupt Status Value Description 0 No interrupt 1 A receive time out has occurred. This bit is cleared by writing a 1 to the RTIC bit in the **UARTICR** register. For receive timeout, the RTIM bit in the **UARTIM** register must be set to see the RTRIS status.
5	TXRIS	RO	0	UART Transmit Raw Interrupt Status Value Description 0 No interrupt 1 If the EOT bit in the **UARTCTL** register is clear, the transmit FIFO level has passed through the condition defined in the **UARTIFLS** register. If the EOT bit is set, the last bit of all transmitted data and flags has left the serializer. This bit is cleared by writing a 1 to the TXIC bit in the **UARTICR** register or by writing data to the transmit FIFO until it becomes greater than the trigger level, if the FIFO is enabled, or by writing a single byte if the FIFO is disabled.
4	RXRIS	RO	0	UART Receive Raw Interrupt Status Value Description 0 No interrupt 1 The receive FIFO level has passed through the condition defined in the **UARTIFLS** register. This bit is cleared by writing a 1 to the RXIC bit in the **UARTICR** register or by reading data from the receive FIFO until it becomes less than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled.

Bit/Field	Name	Type	Reset	Description
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	CTSRIS	RO	0	UART Clear to Send Modem Raw Interrupt Status Value Description 0 No interrupt 1 Clear to Send used for software flow control. This bit is cleared by writing a 1 to the `CTSIC` bit in the **UARTICR** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
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 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x040
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			9BITMIS	reserved	OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	reserved		CTSMIS	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:13	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	9BITMIS	RO	0	9-Bit Mode Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to a receive address match. This bit is cleared by writing a 1 to the `9BITIC` bit in the **UARTICR** register.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEMIS	RO	0	UART Overrun Error Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to an overrun error. This bit is cleared by writing a 1 to the `OEIC` bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
9	BEMIS	RO	0	UART Break Error Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to a break error. This bit is cleared by writing a 1 to the `BEIC` bit in the **UARTICR** register.
8	PEMIS	RO	0	UART Parity Error Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to a parity error. This bit is cleared by writing a 1 to the `PEIC` bit in the **UARTICR** register.
7	FEMIS	RO	0	UART Framing Error Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to a framing error. This bit is cleared by writing a 1 to the `FEIC` bit in the **UARTICR** register.
6	RTMIS	RO	0	UART Receive Time-Out Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to a receive time out. This bit is cleared by writing a 1 to the `RTIC` bit in the **UARTICR** register. For receive timeout, the `RTIM` bit in the **UARTIM** register must be set to see the `RTMIS` status.
5	TXMIS	RO	0	UART Transmit Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 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). This bit is cleared by writing a 1 to the `TXIC` bit in the **UARTICR** register or by writing data to the transmit FIFO until it becomes greater than the trigger level, if the FIFO is enabled, or by writing a single byte if the FIFO is disabled.

Bit/Field	Name	Type	Reset	Description
4	RXMIS	RO	0	UART Receive Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to passing through the specified receive FIFO level. This bit is cleared by writing a 1 to the RXIC bit in the **UARTICR** register or by reading data from the receive FIFO until it becomes less than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled.
3:2	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	CTSMIS	RO	0	UART Clear to Send Modem Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to Clear to Send. This bit is cleared by writing a 1 to the CTSIC bit in the **UARTICR** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 13: UART Interrupt Clear (UARTICR), offset 0x044

The **UARTICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x044
Type W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			9BITIC	reserved	OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	reserved		CTSMIC	reserved
Type	RO	RO	RO	RW	RO	W1C	W1C	W1C	W1C	W1C	W1C	W1C	RO	RO	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
31:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	9BITIC	RW	0	9-Bit Mode Interrupt Clear Writing a 1 to this bit clears the `9BITRIS` bit in the **UARTRIS** register and the `9BITMIS` bit in the **UARTMIS** register.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIC	W1C	0	Overrun Error Interrupt Clear 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.
7	FEIC	W1C	0	Framing Error Interrupt Clear Writing a 1 to this bit clears the `FERIS` bit in the **UARTRIS** register and the `FEMIS` bit in the **UARTMIS** register.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear Writing a 1 to this bit clears the `RTRIS` bit in the **UARTRIS** register and the `RTMIS` bit in the **UARTMIS** register.

Bit/Field	Name	Type	Reset	Description
5	TXIC	W1C	0	Transmit Interrupt Clear Writing a 1 to this bit clears the `TXRIS` bit in the **UARTRIS** register and the `TXMIS` bit in the **UARTMIS** register.
4	RXIC	W1C	0	Receive Interrupt Clear Writing a 1 to this bit clears the `RXRIS` bit in the **UARTRIS** register and the `RXMIS` bit in the **UARTMIS** register.
3:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	CTSMIC	W1C	0	UART Clear to Send Modem Interrupt Clear Writing a 1 to this bit clears the `CTSRIS` bit in the **UARTRIS** register and the `CTSMIS` bit in the **UARTMIS** register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 14: UART DMA Control (UARTDMACTL), offset 0x048

The **UARTDMACTL** register is the DMA control register.

UART DMA Control (UARTDMACTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x048
Type RW, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DMAERR	RW	0	DMA on Error Value Description 0 μDMA receive requests are unaffected when a receive error occurs. 1 μDMA receive requests are automatically disabled when a receive error occurs.
1	TXDMAE	RW	0	Transmit DMA Enable Value Description 0 μDMA for the transmit FIFO is disabled. 1 μDMA for the transmit FIFO is enabled.
0	RXDMAE	RW	0	Receive DMA Enable Value Description 0 μDMA for the receive FIFO is disabled. 1 μDMA for the receive FIFO is enabled.

Register 15: UART 9-Bit Self Address (UART9BITADDR), offset 0x0A4

The **UART9BITADDR** register is used to write the specific address that should be matched with the receiving byte when the 9-bit Address Mask (**UART9BITAMASK**) is set to 0xFF. This register is used in conjunction with **UART9BITAMASK** to form a match for address-byte received.

UART 9-Bit Self Address (UART9BITADDR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x0A4
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	9BITEN	reserved							ADDR							
Type	RW	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	9BITEN	RW	0	Enable 9-Bit Mode Value Description 0 9-bit mode is disabled. 1 9-bit mode is enabled.
14:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ADDR	RW	0x00	Self Address for 9-Bit Mode This field contains the address that should be matched when **UART9BITAMASK** is 0xFF.

Register 16: UART 9-Bit Self Address Mask (UART9BITAMASK), offset 0x0A8

The **UART9BITAMASK** register is used to enable the address mask for 9-bit mode. The address bits are masked to create a set of addresses to be matched with the received address byte.

UART 9-Bit Self Address Mask (UART9BITAMASK)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x0A8
Type RW, reset 0x0000.00FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								MASK							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	MASK	RW	0xFF	Self Address Mask for 9-Bit Mode This field contains the address mask that creates a set of addresses that should be matched.

Register 17: UART Peripheral Properties (UARTPP), offset 0xFC0

The **UARTPP** register provides information regarding the properties of the UART module.

UART Peripheral Properties (UARTPP)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFC0
Type RO, reset 0x0000.0003

	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														NB	SC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

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	NB	RO	0x1	9-Bit Support Value Description 0 The UART module does not provide support for the transmission of 9-bit data for RS-485 support. 1 The UART module provides support for the transmission of 9-bit data for RS-485 support.
0	SC	RO	0x1	Smart Card Support Value Description 0 The UART module does not provide smart card support. 1 The UART module provides smart card support.

Register 18: UART Clock Configuration (UARTCC), offset 0xFC8

The **UARTCC** register controls the baud clock source for the UART module. For more information, see the section called “Communication Clock Sources” on page 211.

Note: If the PIOSC is used for the UART baud clock, the system clock frequency must be at least 9 MHz in Run mode.

UART Clock Configuration (UARTCC)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFC8
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												CS			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	CS	RW	0	UART Baud Clock Source The following table specifies the source that generates for the UART baud clock:

Value	Description
0x0	System clock (based on clock source and divisor factor)
0x1-0x4	reserved
0x5	PIOSC
0x5-0xF	Reserved

Register 19: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 4 (UARTPeriphID4)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFD0
Type RO, reset 0x0000.0000

	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	UART Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 20: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 5 (UARTPeriphID5)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFD4
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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 21: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFD8
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	UART Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 22: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 7 (UARTPeriphID7)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFDC
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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 23: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 0 (UARTPeriphID0)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFE0
Type RO, reset 0x0000.0060

	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 24: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 1 (UARTPeriphID1)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFE4
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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 25: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 2 (UARTPeriphID2)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFE8
Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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 26: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 3 (UARTPeriphID3)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFEC
Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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 27: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 0 (UARTPCellID0)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFF0
Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	UART PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 28: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 1 (UARTPCellID1)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFF4
Type RO, reset 0x0000.00F0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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 29: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 2 (UARTPCellID2)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFF8
Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	UART PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 30: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 3 (UARTPCellID3)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFFC
Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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.

14 Synchronous Serial Interface (SSI)

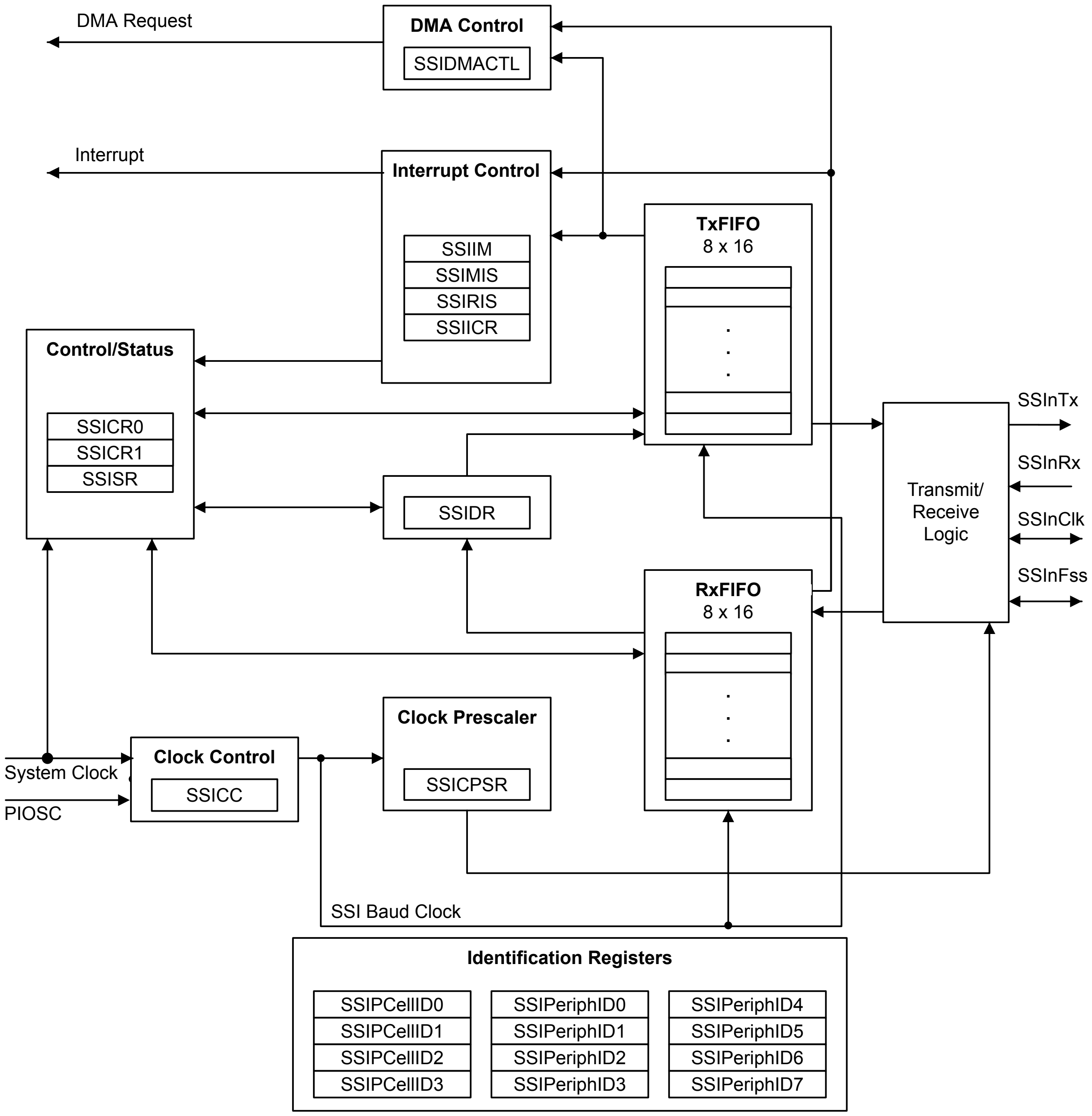
The TM4C1232C3PM microcontroller includes four Synchronous Serial Interface (SSI) modules. Each SSI module is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The TM4C1232C3PM SSI modules have the following features:

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

14.1 Block Diagram

Figure 14-1. SSI Module Block Diagram



14.2 Signal Description

The following table lists the external signals of the SSI module and describes the function of each. Most SSI signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The exceptions to this rule are the SSI0Clk, SSI0Fss, SSI0Rx, and SSI0Tx pins, which default to the SSI function. The "Pin Mux/Pin Assignment" column in the following table lists the possible GPIO pin placements for the SSI signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 603) 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 620) 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 581.

Table 14-1. SSI Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
SSI0Clk	19	PA2 (2)	I/O	TTL	SSI module 0 clock
SSI0Fss	20	PA3 (2)	I/O	TTL	SSI module 0 frame signal
SSI0Rx	21	PA4 (2)	I	TTL	SSI module 0 receive
SSI0Tx	22	PA5 (2)	O	TTL	SSI module 0 transmit
SSI1Clk	30 61	PF2 (2) PD0 (2)	I/O	TTL	SSI module 1 clock.
SSI1Fss	31 62	PF3 (2) PD1 (2)	I/O	TTL	SSI module 1 frame signal.
SSI1Rx	28 63	PF0 (2) PD2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	29 64	PF1 (2) PD3 (2)	O	TTL	SSI module 1 transmit.
SSI2Clk	58	PB4 (2)	I/O	TTL	SSI module 2 clock.
SSI2Fss	57	PB5 (2)	I/O	TTL	SSI module 2 frame signal.
SSI2Rx	1	PB6 (2)	I	TTL	SSI module 2 receive.
SSI2Tx	4	PB7 (2)	O	TTL	SSI module 2 transmit.
SSI3Clk	61	PD0 (1)	I/O	TTL	SSI module 3 clock.
SSI3Fss	62	PD1 (1)	I/O	TTL	SSI module 3 frame signal.
SSI3Rx	63	PD2 (1)	I	TTL	SSI module 3 receive.
SSI3Tx	64	PD3 (1)	O	TTL	SSI module 3 transmit.

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

14.3 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. The SSI also supports the µDMA interface. The transmit and receive FIFOs can be programmed as destination/source addresses in the µDMA module. µDMA operation is enabled by setting the appropriate bit(s) in the **SSIDMACTL** register (see page 912).

14.3.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the input clock (SysClk). The clock is first divided by an even prescale value CPSDVSR from 2 to 254, which is programmed in the **SSI Clock Prescale**

(SSICPSR) register (see page 905). 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 898).

The frequency of the output clock SSInClk is defined by:

```
SSInClk = SysClk / (CPSDVSR * (1 + SCR))
```

Note: The System Clock or the PIOSC can be used as the source for the SSInClk. When the CS field in the **SSI Clock Configuration (SSICC)** register is configured to 0x5, PIOSC is selected as the source. For master mode, the system clock or the PIOSC must be at least two times faster than the SSInClk, with the restriction that SSInClk cannot be faster than 25 MHz. For slave mode, the system clock or the PIOSC must be at least 12 times faster than the SSInClk, with the restriction that SSInClk cannot be faster than 6.67 MHz.

See "Synchronous Serial Interface (SSI)" on page 1154 to view SSI timing parameters.

14.3.2 FIFO Operation

14.3.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 902), 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 SSInTx 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 Rn bit in the **RCGCSSI** register, then 0 is transmitted. Care should be taken to ensure that valid data is in the FIFO as needed. The SSI can be configured to generate an interrupt or a μDMA request when the FIFO is empty.

14.3.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the SSInRx pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

14.3.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service (when the transmit FIFO is half full or less)
- Receive FIFO service (when the receive FIFO is half full or more)
- Receive FIFO time-out
- Receive FIFO overrun
- End of transmission
- Receive DMA transfer complete

- Transmit DMA transfer complete

All of the interrupt events are ORed together before being sent to the interrupt controller, so the 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 906). 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 907 and page 909, respectively).

The receive FIFO has a time-out period that is 32 periods at the rate of SSInClk (whether or not SSInClk is currently active) and is started when the RX FIFO goes from EMPTY to not-EMPTY. If the RX FIFO is emptied before 32 clocks have passed, the time-out period is reset. As a result, the ISR should clear the Receive FIFO Time-out Interrupt just after reading out the RX FIFO by writing a 1 to the RTIC bit in the **SSI Interrupt Clear (SSIICR)** register. The interrupt should not be cleared so late that the ISR returns before the interrupt is actually cleared, or the ISR may be re-activated unnecessarily.

The End-of-Transmission (EOT) interrupt indicates that the data has been transmitted completely and is only valid for Master mode devices/operations. This interrupt can be used to indicate when it is safe to turn off the SSI module clock or enter sleep mode. In addition, because transmitted data and received data complete at exactly the same time, the interrupt can also indicate that read data is ready immediately, without waiting for the receive FIFO time-out period to complete.

Note: In Freescale SPI mode only, a condition can be created where an EOT interrupt is generated for every byte transferred even if the FIFO is full. If the EOT bit has been set to 0 in an integrated slave SSI and the μDMA has been configured to transfer data from this SSI to a Master SSI on the device using external loopback, an EOT interrupt is generated by the SSI slave for every byte even if the FIFO is full.

14.3.4 Frame Formats

Each data frame is between 4 and 16 bits long depending on the size of data programmed and is transmitted starting with the MSB. There are three basic frame types that can be selected by programming the FRF bit in the **SSICR0** register:

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (SSInClk) is held inactive while the SSI is idle, and SSInClk transitions at the programmed frequency only during active transmission or reception of data. The idle state of SSInClk is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI and MICROWIRE frame formats, the serial frame (SSInFss) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

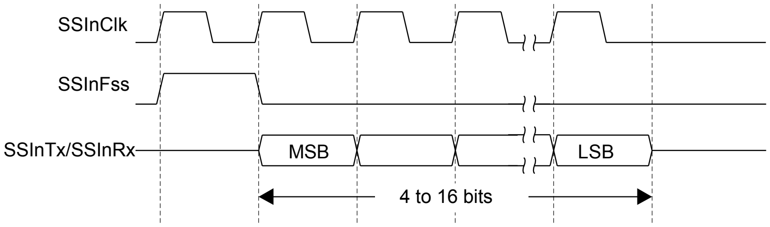
For Texas Instruments synchronous serial frame format, the `SSInFss` pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of `SSInClk` and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the MICROWIRE format uses a special master-slave messaging technique which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

14.3.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 14-2 on page 886 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 14-2. TI Synchronous Serial Frame Format (Single Transfer)

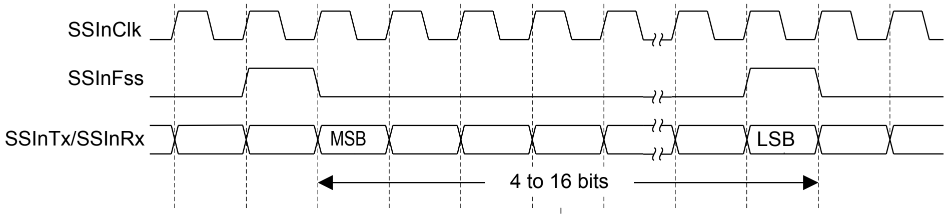


In this mode, `SSInClk` and `SSInFss` are forced Low, and the transmit data line `SSInTx` is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, `SSInFss` is pulsed High for one `SSInClk` period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of `SSInClk`, the MSB of the 4 to 16-bit data frame is shifted out on the `SSInTx` pin. Likewise, the MSB of the received data is shifted onto the `SSInRx` 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 `SSInClk`. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of `SSInClk` after the LSB has been latched.

Figure 14-3 on page 887 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 14-3. TI Synchronous Serial Frame Format (Continuous Transfer)



14.3.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the `SSInFss` signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the `SSInClk` signal are programmable through the `SPO` and `SPH` bits in the **SSICR0** control register.

SPO Clock Polarity Bit

When the `SPO` clock polarity control bit is clear, it produces a steady state Low value on the `SSInClk` pin. If the `SPO` bit is set, a steady state High value is placed on the `SSInClk` pin when data is not being transferred.

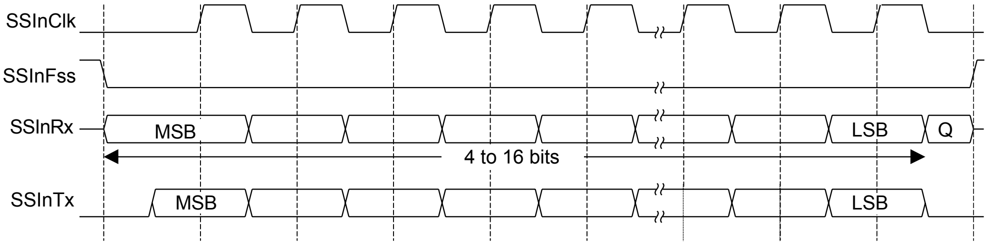
SPH Phase Control Bit

The `SPH` phase control bit selects the clock edge that captures data and allows it to change state. The state of this bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the `SPH` phase control bit is clear, data is captured on the first clock edge transition. If the `SPH` bit is set, data is captured on the second clock edge transition.

14.3.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

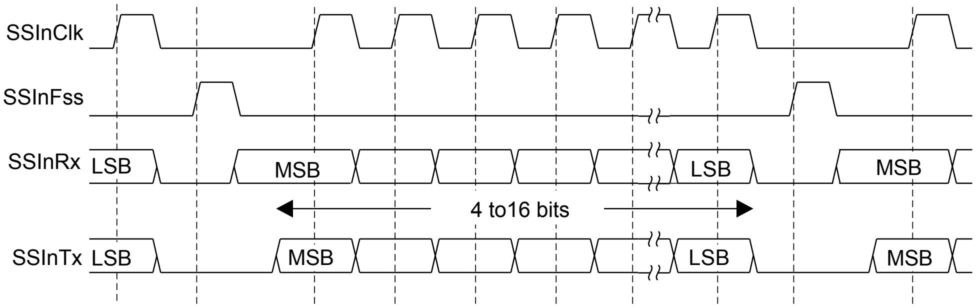
Single and continuous transmission signal sequences for Freescale SPI format with `SPO`=0 and `SPH`=0 are shown in Figure 14-4 on page 888 and Figure 14-5 on page 888.

Figure 14-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0



Note: Q is undefined.

Figure 14-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0



In this configuration, during idle periods:

- SSInClk is forced Low
- SSInFss is forced High
- The transmit data line SSInTx is tristated
- When the SSI is configured as a master, it enables the SSInClk pad
- When the SSI is configured as a slave, it disables the SSInClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven Low, causing slave data to be enabled onto the SSInRx input line of the master. The master SSInTx output pad is enabled.

One half SSInClk period later, valid master data is transferred to the SSInTx pin. Once both the master and slave data have been set, the SSInClk master clock pin goes High after one additional half SSInClk period.

The data is now captured on the rising and propagated on the falling edges of the SSInClk signal.

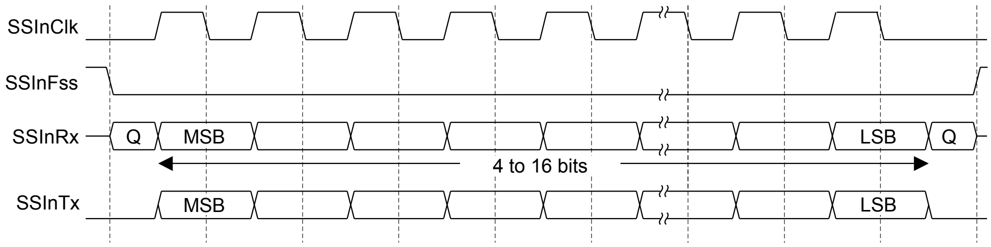
In the case of a single word transmission, after all bits of the data word have been transferred, the SSInFss line is returned to its idle High state one SSInClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSInFss signal must be pulsed High between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is clear. Therefore, the master device must raise the SSInFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSInFss pin is returned to its idle state one SSInClk period after the last bit has been captured.

14.3.4.4 Freescale SPI Frame Format with SPO=0 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=0 and SPH=1 is shown in Figure 14-6 on page 889, which covers both single and continuous transfers.

Figure 14-6. Freescale SPI Frame Format with SPO=0 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

- SSInClk is forced Low
- SSInFss is forced High
- The transmit data line SSInTx is tristated
- When the SSI is configured as a master, it enables the SSInClk pad
- When the SSI is configured as a slave, it disables the SSInClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven Low. The master SSInTx output is enabled. After an additional one-half SSInClk period, both master and slave valid data are enabled onto their respective transmission lines. At the same time, the SSInClk is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSInClk signal.

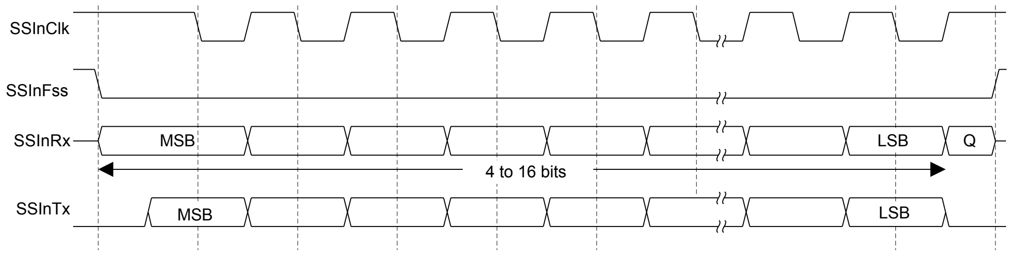
In the case of a single word transfer, after all bits have been transferred, the SSInFss line is returned to its idle High state one SSInClk period after the last bit has been captured.

For continuous back-to-back transfers, the SSInFss pin is held Low between successive data words, and termination is the same as that of the single word transfer.

14.3.4.5 Freescale SPI Frame Format with SPO=1 and SPH=0

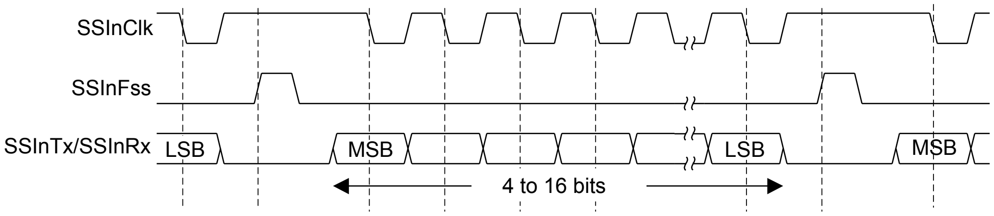
Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 14-7 on page 890 and Figure 14-8 on page 890.

Figure 14-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0



Note: Q is undefined.

Figure 14-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0



In this configuration, during idle periods:

- SSInClk is forced High
- SSInFss is forced High
- The transmit data line SSInTx is tristated
- When the SSI is configured as a master, it enables the SSInClk pad
- When the SSI is configured as a slave, it disables the SSInClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven Low, causing slave data to be immediately transferred onto the SSInRx line of the master. The master SSInTx output pad is enabled.

One-half period later, valid master data is transferred to the SSInTx line. Once both the master and slave data have been set, the SSInClk master clock pin becomes Low after one additional half

SSInClk period, meaning that data is captured on the falling edges and propagated on the rising edges of the SSInClk signal.

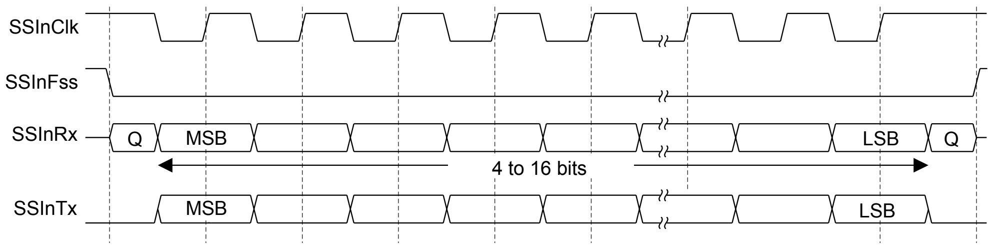
In the case of a single word transmission, after all bits of the data word are transferred, the SSInFss line is returned to its idle High state one SSInClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSInFss signal must be pulsed High between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is clear. Therefore, the master device must raise the SSInFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSInFss pin is returned to its idle state one SSInClk period after the last bit has been captured.

14.3.4.6 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 14-9 on page 891, which covers both single and continuous transfers.

Figure 14-9. Freescale SPI Frame Format with SPO=1 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

- SSInClk is forced High
- SSInFss is forced High
- The transmit data line SSInTx is tristated
- When the SSI is configured as a master, it enables the SSInClk pad
- When the SSI is configured as a slave, it disables the SSInClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven Low. The master SSInTx output pad is enabled. After an additional one-half SSInClk period, both master and slave data are enabled onto their respective transmission lines. At the same time, SSInClk is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSInClk signal.

After all bits have been transferred, in the case of a single word transmission, the SSInFss line is returned to its idle high state one SSInClk period after the last bit has been captured.

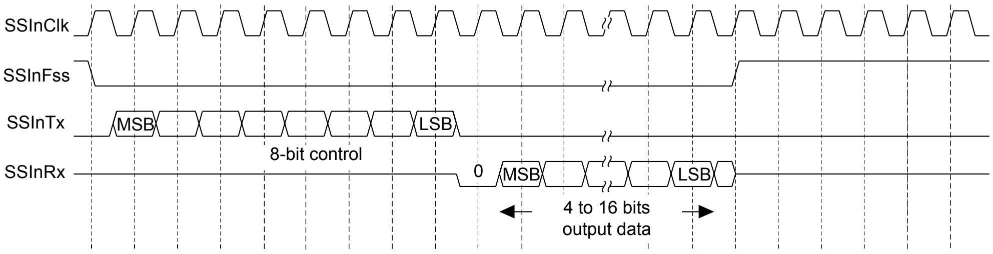
For continuous back-to-back transmissions, the `SSInFss` pin remains in its active Low state until the final bit of the last word has been captured and then returns to its idle state as described above.

For continuous back-to-back transfers, the `SSInFss` pin is held Low between successive data words and termination is the same as that of the single word transfer.

14.3.4.7 MICROWIRE Frame Format

Figure 14-10 on page 892 shows the MICROWIRE frame format for a single frame. Figure 14-11 on page 893 shows the same format when back-to-back frames are transmitted.

Figure 14-10. MICROWIRE Frame Format (Single Frame)



MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex and uses a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSI to the off-chip slave device. During this transmission, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- `SSInClk` is forced Low
- `SSInFss` is forced High
- The transmit data line `SSInTx` is tristated

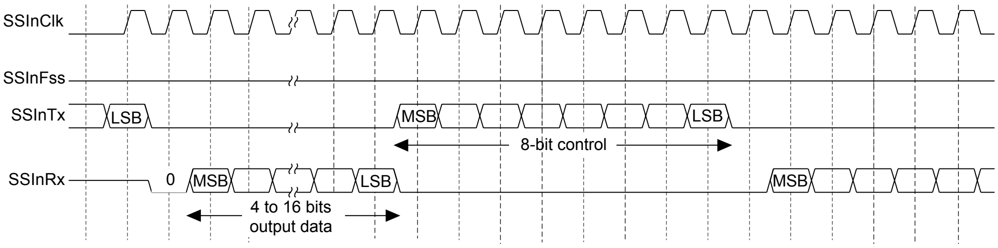
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of `SSInFss` causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic and the MSB of the 8-bit control frame to be shifted out onto the `SSInTx` pin. `SSInFss` remains Low for the duration of the frame transmission. The `SSInRx` 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 `SSInClk`. 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 `SSInRx` line on the falling edge of `SSInClk`. The SSI in turn latches each bit on the rising edge of `SSInClk`. At the end of the frame, for single transfers, the `SSInFss` 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 SSInClk after the LSB has been latched by the receive shifter or when the SSInFss pin goes High.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSInFss 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 SSInClk, after the LSB of the frame has been latched into the SSI.

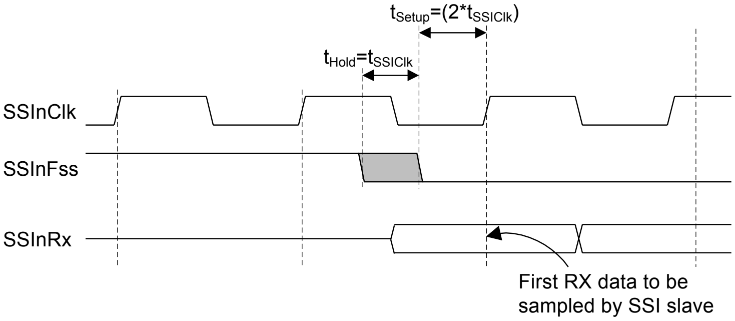
Figure 14-11. MICROWIRE Frame Format (Continuous Transfer)



In the MICROWIRE mode, the SSI slave samples the first bit of receive data on the rising edge of SSInClk after SSInFss has gone Low. Masters that drive a free-running SSInClk must ensure that the SSInFss signal has sufficient setup and hold margins with respect to the rising edge of SSInClk.

Figure 14-12 on page 893 illustrates these setup and hold time requirements. With respect to the SSInClk rising edge on which the first bit of receive data is to be sampled by the SSI slave, SSInFss must have a setup of at least two times the period of SSInClk on which the SSI operates. With respect to the SSInClk rising edge previous to this edge, SSInFss must have a hold of at least one SSInClk period.

Figure 14-12. MICROWIRE Frame Format, SSInFss Input Setup and Hold Requirements


t_{Setup}=(2*t_{SSIClk})
t_{Hold}=t_{SSIClk}
SSInClk
SSInFss
SSInRx
First RX data to be sampled by SSI slave

14.3.5 DMA Operation

The SSI peripheral provides an interface to the μDMA controller with separate channels for transmit and receive. The μDMA operation of the SSI is enabled through the **SSI DMA Control (SSIDMACTL)** register. When μDMA operation is enabled, the SSI asserts a μDMA request on the receive or transmit channel when the associated FIFO can transfer data.

For the receive channel, a single transfer request is asserted whenever any data is in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is 4 or more items. For the transmit channel, a single transfer request is asserted whenever at least one empty location is in the transmit FIFO. The burst request is asserted whenever the transmit FIFO has 4 or more empty slots. The single and burst μDMA transfer requests are handled automatically by the μDMA controller depending how the μDMA channel is configured.

To enable μDMA operation for the receive channel, the RXDMAE bit of the **DMA Control (SSIDMACTL)** register should be set after configuring the μDMA. To enable μDMA operation for the transmit channel, the TXDMAE bit of **SSIDMACTL** should be set after configuring the μDMA. If μDMA is enabled, then the μDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the SSI interrupt vector. Therefore, if interrupts are used for SSI operation and μDMA is enabled, the SSI interrupt handler must be designed to handle the μDMA completion interrupt.

When transfers are performed from a FIFO of the SSI using the μDMA, and any interrupt is generated from the SSI, the SSI module's status bit in the **DMA Channel Interrupt Status (DMACHIS)** register must be checked at the end of the interrupt service routine. If the status bit is set, clear the interrupt by writing a 1 to it.

See “Micro Direct Memory Access (μDMA)” on page 517 for more details about programming the μDMA controller.

14.4 Initialization and Configuration

To enable and initialize the SSI, the following steps are necessary:

1. Enable the SSI module using the **RCGCSSI** register (see page 324).

2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register (see page 319). To find out which GPIO port to enable, refer to Table 20-5 on page 1115.

3. Set the GPIO AFSEL bits for the appropriate pins (see page 603). To determine which GPIOs to configure, see Table 20-4 on page 1111.

4. Configure the PMCn fields in the **GPIOPCTL** register to assign the SSI signals to the appropriate pins. See page 620 and Table 20-5 on page 1115.

5. Program the **GPIODEN** register to enable the pin's digital function. In addition, the drive strength, drain select and pull-up/pull-down functions must be configured. Refer to “General-Purpose Input/Outputs (GPIOs)” on page 581 for more information.

   Note: Pull-ups can be used to avoid unnecessary toggles on the SSI pins, which can take the slave to a wrong state. In addition, if the SSIClk signal is programmed to steady state High through the SPO bit in the **SSICR0** register, then software must also configure the GPIO port pin corresponding to the SSInClk signal as a pull-up in the **GPIO Pull-Up Select (GPIOPUR)** register.

For each of the frame formats, the 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 SSI clock source by writing to the **SSICC** register.

4. Configure the clock prescale divisor by writing the **SSICPSR** register.

5. Write the **SSICR0** register with the following configuration:

   - Serial clock rate (SCR)
   - Desired clock phase/polarity, if using Freescale SPI mode (SPH and SPO)
   - The protocol mode: Freescale SPI, TI SSF, MICROWIRE (FRF)
   - The data size (DSS)

6. Optionally, configure the SSI module for µDMA use with the following steps:

   a. Configure a µDMA for SSI use. See "Micro Direct Memory Access (µDMA)" on page 517 for more information.

   b. Enable the SSI Module's TX FIFO or RX FIFO by setting the TXDMAE or RXDMAE bit in the **SSIDMACTL** register.

7. 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:

$$SSInClk = SysClk / (CPSDVSR * (1 + SCR))$$
$$1x10^6 = 20x10^6 / (CPSDVSR * (1 + SCR))$$

In this case, if CPSDVSR=0x2, SCR must be 0x9.

The configuration sequence would be as follows:

1. Ensure that the SSE bit in the **SSICR1** register is clear.

2. Write the **SSICR1** register with a value of 0x0000.0000.

3. Write the **SSICPSR** register with a value of 0x0000.0002.

4. Write the **SSICR0** register with a value of 0x0000.09C7.

5. The SSI is then enabled by setting the SSE bit in the **SSICR1** register.

14.5 Register Map

Table 14-2 on page 896 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
- SSI2: 0x4000.A000
- SSI3: 0x4000.B000

Note that the SSI module clock must be enabled before the registers can be programmed (see page 324). The Rn bit of the **PRSSI** register must be read as 0x1 before any SSI module registers are accessed.

Note: The SSI must be disabled (see the SSE bit in the **SSICR1** register) before any of the control registers are reprogrammed.

Table 14-2. SSI Register Map

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	RW	0x0000.0000	SSI Control 0	898
0x004	SSICR1	RW	0x0000.0000	SSI Control 1	900
0x008	SSIDR	RW	0x0000.0000	SSI Data	902
0x00C	SSISR	RO	0x0000.0003	SSI Status	903
0x010	SSICPSR	RW	0x0000.0000	SSI Clock Prescale	905
0x014	SSIIM	RW	0x0000.0000	SSI Interrupt Mask	906
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	907
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	909
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	911
0x024	SSIDMACTL	RW	0x0000.0000	SSI DMA Control	912
0xFC8	SSICC	RW	0x0000.0000	SSI Clock Configuration	913
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	914
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	915
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	916
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	917
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	918
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	919
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	920
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	921
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	922
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	923

Table 14-2. SSI Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	924
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	925

14.6 Register Descriptions

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

Register 1: SSI Control 0 (SSICR0), offset 0x000

The **SSICR0** register contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x000
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SCR								SPH	SPO	FRF		DSS			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	SCR	RW	0x00	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=SysClk/(CPSDVSR * (1 + SCR))` where `CPSDVSR` is an even value from 2-254 programmed in the **SSICPSR** register, and `SCR` is a value from 0-255.
7	SPH	RW	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	RW	0	SSI Serial Clock Polarity Value Description 0 A steady state Low value is placed on the `SSInClk` pin. 1 A steady state High value is placed on the `SSInClk` pin when data is not being transferred. **Note:** If this bit is set, then software must also configure the GPIO port pin corresponding to the `SSInClk` signal as a pull-up in the **GPIO Pull-Up Select (GPIOPUR)** register.

Bit/Field	Name	Type	Reset	Description
5:4	FRF	RW	0x0	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	RW	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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x004
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											EOT	reserved	MS	SSE	LBM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:5	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	RW	0	End of Transmission This bit is only valid for Master mode devices and operations (MS = 0x0). 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. **Note:** In Freescale SPI mode only, a condition can be created where an EOT interrupt is generated for every byte transferred even if the FIFO is full. If the EOT bit has been set to 0 in an integrated slave SSI and the µDMA has been configured to transfer data from this SSI to a Master SSI on the device using external loopback, an EOT interrupt is generated by the SSI slave for every byte even if the FIFO is full.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	MS	RW	0	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	RW	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	RW	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

Important: This register is read-sensitive. See the register description for details.

The **SSIDR** register is 16-bits wide. When the **SSIDR** register is read, the entry in the receive FIFO that is pointed to by the current FIFO read pointer is accessed. When a data value is removed by the SSI receive logic from the incoming data frame, it is placed into the entry in the receive FIFO pointed to by the current FIFO write pointer.

When the **SSIDR** register is written to, the entry in the transmit FIFO that is pointed to by the write pointer is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. Each data value is loaded into the transmit serial shifter, then serially shifted out onto the `SSInTx` pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x008
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	RW	0x0000	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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x00C
Type RO, reset 0x0000.0003

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											BSY	RFF	RNE	TNF	TFE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	BSY	RO	0	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	RO	1	SSI Transmit FIFO Empty Value Description 0 The transmit FIFO is not empty. 1 The transmit FIFO is empty.

Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

The **SSICPSR** register specifies the division factor which is used to derive the `SSInClk` from the system clock. The clock is further divided by a value from 1 to 256, which is 1 + `SCR`. `SCR` is programmed in the **SSICR0** register. The frequency of the `SSInClk` is defined by:

```
SSInClk = SysClk / (CPSDVSR * (1 + SCR))
```

The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

SSI Clock Prescale (SSICPSR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x010
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CPSDVSR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CPSDVSR	RW	0x00	SSI Clock Prescale Divisor This value must be an even number from 2 to 254, depending on the frequency of `SSInClk`. The LSB always returns 0 on reads.

Register 6: 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 clears the mask, enabling the interrupt to be sent to the interrupt controller. Clearing a bit sets the corresponding mask, preventing the interrupt from being signaled to the controller.

SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x014
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												TXIM	RXIM	RTIM	RORIM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXIM	RW	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	RW	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	RW	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	RW	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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x018
Type RO, reset 0x0000.0008

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												TXRIS	RXRIS	RTRIS	RORRIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of 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 empty 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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	0	Software should not rely on the value of 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 empty or less (if the `EOT` bit is clear) or due to the transmission of the last data bit (if the `EOT` bit is set). This bit is cleared when the transmit FIFO is more than half empty (if the `EOT` bit is clear) or when it has any data in it (if the `EOT` bit is set).
2	RXMIS	RO	0	SSI Receive FIFO Masked Interrupt Status 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 more. 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 `RTIC` bit in the **SSI Interrupt Clear (SSIICR)** register.

Bit/Field	Name	Type	Reset	Description
0	RORMIS	RO	0	SSI Receive Overrun Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the receive FIFO overflowing. This bit is cleared when a 1 is written to the `RORIC` bit in the **SSI Interrupt Clear (SSIICR)** register.

Register 9: SSI Interrupt Clear (SSIICR), offset 0x020

The **SSIICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

SSI Interrupt Clear (SSIICR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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	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
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RTIC	W1C	0	SSI Receive Time-Out Interrupt Clear 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 μDMA control register.

SSI DMA Control (SSIDMACTL)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x024
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														TXDMAE	RXDMAE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXDMAE	RW	0	Transmit DMA Enable Value Description 0 μDMA for the transmit FIFO is disabled. 1 μDMA for the transmit FIFO is enabled.
0	RXDMAE	RW	0	Receive DMA Enable Value Description 0 μDMA for the receive FIFO is disabled. 1 μDMA for the receive FIFO is enabled.

Register 11: SSI Clock Configuration (SSICC), offset 0xFC8

The **SSICC** register controls the baud clock source for the SSI module.

Note: If the PIOSC is used for the SSI baud clock, the system clock frequency must be at least 16 MHz in Run mode.

SSI Clock Configuration (SSICC)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0xFC8
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												CS			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	CS	RW	0	SSI Baud Clock Source The following table specifies the source that generates for the SSI baud clock:

Value	Description
0x0	System clock (based on clock source and divisor factor)
0x1-0x4	reserved
0x5	PIOSC
0x6 - 0xF	Reserved

Register 12: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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 13: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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 14: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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 15: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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 16: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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 17: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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 18: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 19: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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 20: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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 21: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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 22: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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 23: 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
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
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.

15 Inter-Integrated Circuit (I^{2}C) Interface

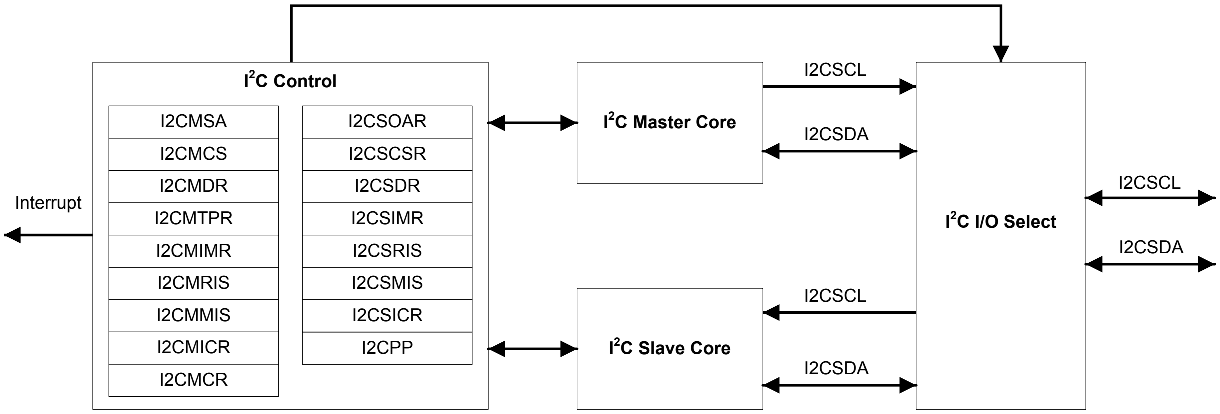
The Inter-Integrated Circuit (I^{2}C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I^{2}C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I^{2}C bus may also be used for system testing and diagnostic purposes in product development and manufacturing. The TM4C1232C3PM microcontroller includes providing the ability to communicate (both transmit and receive) with other I^{2}C devices on the bus.

The TM4C1232C3PM controller includes I^{2}C modules with the following features:

- Devices on the I^{2}C bus can be designated as either a master or a slave
  - Supports both transmitting and receiving data as either a master or a slave
  - Supports simultaneous master and slave operation
- Four I^{2}C modes
  - Master transmit
  - Master receive
  - Slave transmit
  - Slave receive
- Four transmission speeds:
  - Standard (100 Kbps)
  - Fast-mode (400 Kbps)
  - Fast-mode plus (1 Mbps)
  - High-speed mode (3.33 Mbps)
- Clock low timeout interrupt
- Dual slave address capability
- Glitch suppression
- Master and slave interrupt generation
  - Master generates interrupts when a transmit or receive operation completes (or aborts due to an error)
  - Slave generates interrupts when data has been transferred or requested by a master or when a START or STOP condition is detected
- Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

15.1 Block Diagram

Figure 15-1. I²C Block Diagram



15.2 Signal Description

The following table lists the external signals of the I²C interface and describes the function of each. The I²C interface signals are alternate functions for some GPIO signals and default to be GPIO signals at reset, with the exception of the `I2C0SCL` and `I2CSDA` pins which default to the I²C function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the I²C signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 603) should be set to choose the I²C function. The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPCTL)** register (page 620) to assign the I²C signal to the specified GPIO port pin. Note that the `I2CSDA` pin should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 581.

Table 15-1. I2C Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
I2C0SCL	47	PB2 (3)	I/O	OD	I²C module 0 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C0SDA	48	PB3 (3)	I/O	OD	I²C module 0 data.
I2C1SCL	23 33	PA6 (3) PG4 (3)	I/O	OD	I²C module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C1SDA	24 32	PA7 (3) PG5 (3)	I/O	OD	I²C module 1 data.
I2C2SCL	59	PE4 (3)	I/O	OD	I²C module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C2SDA	60	PE5 (3)	I/O	OD	I²C module 2 data.
I2C3SCL	37 61	PG0 (3) PD0 (3)	I/O	OD	I²C module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.

Table 15-1. I2C Signals (64LQFP) ***(continued)***

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`I2C3SDA`	36 62	PG1 (3) PD1 (3)	I/O	OD	I^{2}C module 3 data.
`I2C4SCL`	35	PG2 (3)	I/O	OD	I^{2}C module 4 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
`I2C4SDA`	34	PG3 (3)	I/O	OD	I^{2}C module 4 data.
`I2C5SCL`	1	PB6 (3)	I/O	OD	I^{2}C module 5 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
`I2C5SDA`	4	PB7 (3)	I/O	OD	I^{2}C module 5 data.

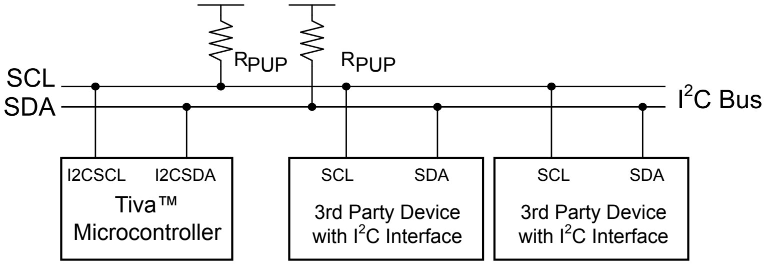
[a] The TTL designation indicates the pin has TTL-compatible voltage levels.

15.3 Functional Description

Each I^{2}C module is comprised of both master and slave functions and is identified by a unique address. A master-initiated communication generates the clock signal, SCL. For proper operation, the SDA pin must be configured as an open-drain signal. Due to the internal circuitry that supports high-speed operation, the SCL pin must not be configured as an open-drain signal, although the internal circuitry causes it to act as if it were an open drain signal. Both SDA and SCL signals must be connected to a positive supply voltage using a pull-up resistor. A typical I^{2}C bus configuration is shown in Figure 15-2. Refer to the *I2C-bus specification and user manual* to determine the size of the pull-ups needed for proper operation.

See "Inter-Integrated Circuit (I^{2}C) Interface" on page 1157 for I^{2}C timing diagrams.

Figure 15-2. I^{2}C Bus Configuration



15.3.1 I^{2}C Bus Functional Overview

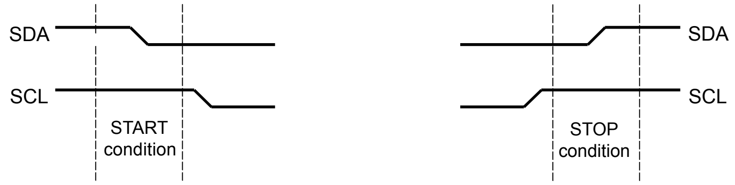
The I^{2}C bus uses only two signals: SDA and SCL, named `I2CSDA` and `I2CSCL` on TM4C1232C3PM microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are High.

Every transaction on the I^{2}C 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 929) is unrestricted, but each data byte has to be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it can hold the clock line SCL Low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.

15.3.1.1 START and STOP Conditions

The protocol of the I^{2}C bus defines two states to begin and end a transaction: START and STOP. A High-to-Low transition on the SDA line while the SCL is High is defined as a START condition, and a Low-to-High transition on the SDA line while SCL is High is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 15-3.

Figure 15-3. START and STOP Conditions



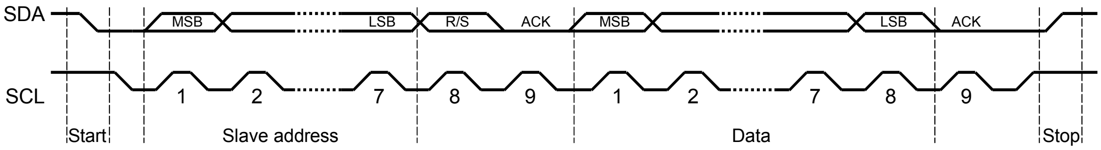
The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the **I^{2}C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is cleared, and the Control register is written 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 **I^{2}C Master Data (I2CMDR)** register. When the I^{2}C module operates in Master receiver mode, the ACK bit is normally set causing the I^{2}C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I^{2}C bus controller requires no further data to be transmitted from the slave transmitter.

When operating in slave mode, the STARTRIS and STOPRIS bits in the **I^{2}C Slave Raw Interrupt Status (I2CSRIS)** register indicate detection of start and stop conditions on the bus and the **I^{2}C Slave Masked Interrupt Status (I2CSMIS)** register can be configured to allow STARTRIS and STOPRIS to be promoted to controller interrupts (when interrupts are enabled).

15.3.1.2 Data Format with 7-Bit Address

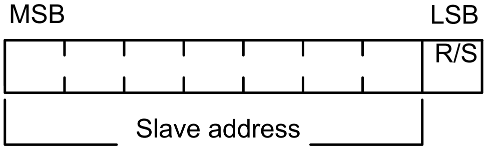
Data transfers follow the format shown in Figure 15-4. After the START condition, a slave address is transmitted. This address is 7-bits long followed by an eighth bit, which is a data direction bit (R/S bit in the **I2CMSA** register). If the R/S bit is clear, it indicates a transmit operation (send), and if it is set, it indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition. Various combinations of receive/transmit formats are then possible within a single transfer.

Figure 15-4. Complete Data Transfer with a 7-Bit Address



The first seven bits of the first byte make up the slave address (see Figure 15-5). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master transmits (sends) data to the selected slave, and a one in this position means that the master receives data from the slave.

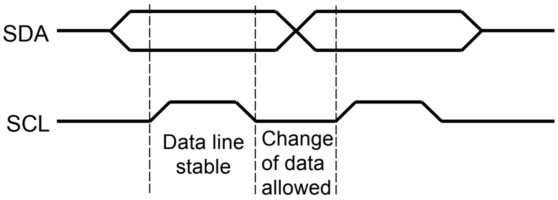
Figure 15-5. R/S Bit in First Byte



15.3.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is Low (see Figure 15-6).

Figure 15-6. Data Validity During Bit Transfer on the I^2C Bus



15.3.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data transmitted out by the receiver during the acknowledge cycle must comply with the data validity requirements described in “Data Validity” on page 930.

When a slave receiver does not acknowledge the slave address, SDA must be left High by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Because the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

If the slave is required to provide a manual ACK or NACK, the **I^2C Slave ACK Control (I2CSACKCTL)** register allows the slave to NACK for invalid data or command or ACK for valid data or command. When this operation is enabled, the MCU slave module I^2C clock is pulled low after the last data bit until this register is written with the indicated response.

15.3.1.5 Repeated Start

The I^2C master module has the capability of executing a repeated START (transmit or receive) after an initial transfer has occurred.

A repeated start sequence for a Master transmit is as follows:

1. When the device is in the idle state, the Master writes the slave address to the **I2CMSA** register and configures the R/S bit for the desired transfer type.

2. Data is written to the **I2CMDR** register.

3. When the BUSY bit in the **I2CMCS** register is 0 , the Master writes 0x3 to the **I2CMCS** register to initiate a transfer.

4. The Master does not generate a STOP condition but instead writes another slave address to the **I2CMSA** register and then writes 0x3 to initiate the repeated START.

A repeated start sequence for a Master receive is similar:

1. When the device is in idle, the Master writes the slave address to the **I2CMSA** register and configures the R/S bit for the desired transfer type.

2. The master reads data from the **I2CMDR** register.

3. When the BUSY bit in the **I2CMCS** register is 0 , the Master writes 0x3 to the **I2CMCS** register to initiate a transfer.

4. The Master does not generate a STOP condition but instead writes another slave address to the **I2CMSA** register and then writes 0x3 to initiate the repeated START.

For more information on repeated START, refer to Figure 15-12 on page 941 and Figure 15-13 on page 942.

15.3.1.6 Clock Low Timeout (CLTO)

The I^{2}C slave can extend the transaction by pulling the clock low periodically to create a slow bit transfer rate. The I^{2}C module has a 12-bit programmable counter that is used to track how long the clock has been held low. The upper 8 bits of the count value are software programmable through the **I^{2}C Master Clock Low Timeout Count (I2CMCLKOCNT)** register. The lower four bits are not user visible and are 0x0. The CNTL value programmed in the **I2CMCLKOCNT** register has to be greater than 0x01. The application can program the eight most significant bits of the counter to reflect the acceptable cumulative low period in transaction. The count is loaded at the START condition and counts down on each falling edge of the internal bus clock of the Master. Note that the internal bus clock generated for this counter keeps running at the programmed I^{2}C speed even if SCL is held low on the bus. Upon reaching terminal count, the master state machine forces ABORT on the bus by issuing a STOP condition at the instance of SCL and SDA release.

As an example, if an I^{2}C module was operating at 100 kHz speed, programming the **I2CMCLKOCNT** register to 0xDA would translate to the value 0xDA0 since the lower four bits are set to 0x0. This would translate to a decimal value of 3488 clocks or a cumulative clock low period of 34.88 ms at 100 kHz.

The CLKRIS bit in the **I^{2}C Master Raw Interrupt Status (I2CMRIS)** register is set when the clock timeout period is reached, allowing the master to start corrective action to resolve the remote slave state. In addition, the CLKTO bit in the **I^{2}C Master Control/Status (I2CMCS)** register is set; this bit is cleared when a STOP condition is sent or during the I^{2}C master reset. The status of the raw SDA and SCL signals are readable by software through the SDA and SCL bits in the **I^{2}C Master Bus Monitor (I2CMBMON)** register to help determine the state of the remote slave.

In the event of a CLTO condition, application software must choose how it intends to attempt bus recovery. Most applications may attempt to manually toggle the I^{2}C pins to force the slave to let go of the clock signal (a common solution is to attempt to force a STOP on the bus). If a CLTO is detected before the end of a burst transfer, and the bus is successfully recovered by the master, the master hardware attempts to finish the pending burst operation. Depending on the state of the slave after bus recovery, the actual behavior on the bus varies. If the slave resumes in a state where it can acknowledge the master (essentially, where it was before the bus hang), it continues where it left off. However, if the slave resumes in a reset state (or if a forced STOP by the master causes the slave to enter the idle state), it may ignore the master's attempt to complete the burst operation and NAK the first data byte that the master sends or requests.

Since the behavior of slaves cannot always be predicted, it is suggested that the application software always write the STOP bit in the **I^2C Master Configuration (I2CMCR)** register during the CLTO interrupt service routine. This limits the amount of data the master attempts to send or receive upon bus recovery to a single byte, and after the single byte is on the wire, the master issues a STOP. An alternative solution is to have the application software reset the I^2C peripheral before attempting to manually recover the bus. This solution allows the I^2C master hardware to be returned to a known good (and idle) state before attempting to recover a stuck bus and prevents any unwanted data from appearing on the wire.

Note: The Master Clock Low Timeout counter counts for the entire time SCL is held Low continuously. If SCL is deasserted at any point, the Master Clock Low Timeout Counter is reloaded with the value in the **I2CMCLKOCNT** register and begins counting down from this value.

15.3.1.7 Dual Address

The I^2C interface supports dual address capability for the slave. The additional programmable address is provided and can be matched if enabled. In legacy mode with dual address disabled, the I^2C slave provides an ACK on the bus if the address matches the OAR field in the **I2CSOAR** register. In dual address mode, the I^2C slave provides an ACK on the bus if either the OAR field in the **I2CSOAR** register or the OAR2 field in the **I2CSOAR2** register is matched. The enable for dual address is programmable through the OAR2EN bit in the **I2CSOAR2** register and there is no disable on the legacy address.

The OAR2SEL bit in the **I2CSCSR** register indicates if the address that was ACKed is the alternate address or not. When this bit is clear, it indicates either legacy operation or no address match.

15.3.1.8 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is High. During arbitration, the first of the competing master devices to place a 1 (High) on SDA, while another master transmits a 0 (Low), switches off its data output stage and retires until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

15.3.1.9 Glitch Suppression in Multi-Master Configuration

When a multi-master configuration is being used, the GFE bit in the **I^2C Master Configuration (I2CMCR)** register can be set to enable glitch suppression on the SCL and SDA lines and assure proper signal values. The filter can be programmed to different filter widths using the GFPW bit in the **I^2C Master Configuration 2 (I2CMCR2)** register. The glitch suppression value is in terms of buffered system clocks. Note that all signals will be delayed internally when glitch suppression is nonzero. For example, if GFPW is set to 0x7, 31 clocks should be added onto the calculation for the expected transaction time.

15.3.2 Available Speed Modes

The I^{2}C bus can run in Standard mode (100 kbps), Fast mode (400 kbps), Fast mode plus (1 Mbps) or High-Speed mode (3.33 Mbps). The selected mode should match the speed of the other I^{2}C devices on the bus.

15.3.2.1 Standard, Fast, and Fast Plus Modes

Standard, Fast, and Fast Plus modes are selected using a value in the **I^{2}C Master Timer Period (I2CMTPR)** register that results in an SCL frequency of 100 kbps for Standard mode, 400 kbps for Fast mode, or 1 Mbps for Fast mode plus.

The I^{2}C 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 955). This value is determined by replacing the known variables in the equation below and solving for *TIMER_PRD*.

The I^{2}C clock period is calculated as follows:

$$SCL\_PERIOD = 2 \times (1 + TIMER\_PRD) \times (SCL\_LP + SCL\_HP) \times CLK\_PRD$$

For example:

$CLK\_PRD = 50$ ns

$TIMER\_PRD = 2$

$SCL\_LP=6$

$SCL\_HP=4$

yields a SCL frequency of:

$1/SCL\_PERIOD = 333$ Khz

Table 15-2 gives examples of the timer periods that should be used to generate Standard, Fast mode, and Fast mode plus SCL frequencies based on various system clock frequencies.

Table 15-2. Examples of I^{2}C Master Timer Period Versus Speed Mode

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode	Timer Period	Fast Mode Plus
4 MHz	0x01	100 Kbps	-	-	-	-
6 MHz	0x02	100 Kbps	-	-	-	-
12.5 MHz	0x06	89 Kbps	0x01	312 Kbps	-	-
16.7 MHz	0x08	93 Kbps	0x02	278 Kbps	-	-
20 MHz	0x09	100 Kbps	0x02	333 Kbps	-	-
25 MHz	0x0C	96.2 Kbps	0x03	312 Kbps	-	-
33 MHz	0x10	97.1 Kbps	0x04	330 Kbps	-	-
40 MHz	0x13	100 Kbps	0x04	400 Kbps	0x01	1000 Kbps
50 MHz	0x18	100 Kbps	0x06	357 Kbps	0x02	833 Kbps
80 MHz	0x27	100 Kbps	0x09	400 Kbps	0x03	1000 Kbps

15.3.2.2 High-Speed Mode

The TM4C1232C3PM I^2C peripheral has support for High-speed operation as both a master and slave. High-Speed mode is configured by setting the HS bit in the **I^2C Master Control/Status (I2CMCS)** register. High-Speed mode transmits data at a high bit rate with a 66.6%/33.3% duty cycle, but communication and arbitration are done at Standard, Fast mode, or Fast-mode plus speed, depending on which is selected by the user. When the HS bit in the **I2CMCS** register is set, current mode pull-ups are enabled.

The clock period can be selected using the equation below, but in this case, SCL_LP=2 and SCL_HP=1.

$$SCL\_PERIOD = 2 \times (1 + TIMER\_PRD) \times (SCL\_LP + SCL\_HP) \times CLK\_PRD$$

So for example:

```
CLK_PRD = 25 ns
TIMER_PRD = 1
SCL_LP=2
SCL_HP=1
```

yields a SCL frequency of:

```
1/T = 3.33 Mhz
```

Table 15-3 on page 934 gives examples of timer period and system clock in High-Speed mode. Note that the HS bit in the **I2CMTPR** register needs to be set for the TPR value to be used in High-Speed mode.

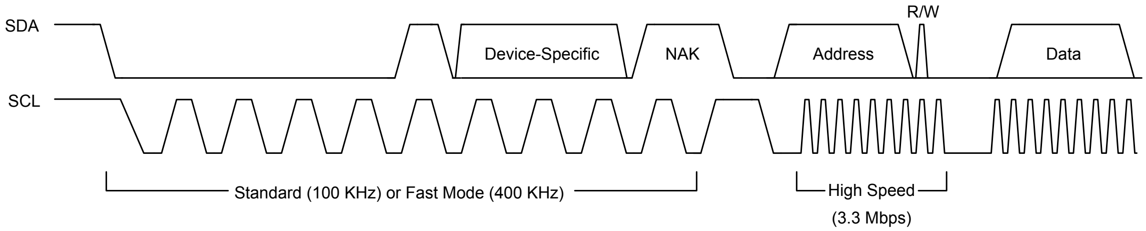
Table 15-3. Examples of I^2C Master Timer Period in High-Speed Mode

System Clock	Timer Period	Transmission Mode
40 MHz	0x01	3.33 Mbps
50 MHz	0x02	2.77 Mbps
80 MHz	0x03	3.33 Mbps

When operating as a master, the protocol is shown in Figure 15-7. The master is responsible for sending a master code byte in either Standard (100 Kbps) or Fast-mode (400 Kbps) before it begins transferring in High-speed mode. The master code byte must contain data in the form of 0000.1XXX and is used to tell the slave devices to prepare for a High-speed transfer. The master code byte should never be acknowledged by a slave since it is only used to indicate that the upcoming data is going to be transferred at a higher data rate. To send the master code byte, software should place the value of the master code byte into the **I2CMSA** register and write the **I2CMCS** register with a value of 0x13. This places the I^2C master peripheral in High-speed mode, and all subsequent transfers (until STOP) are carried out at High-speed data rate using the normal **I2CMCS** command bits, without setting the HS bit in the **I2CMCS** register. Again, setting the HS bit in the **I2CMCS** register is only necessary during the master code byte.

When operating as a High-speed slave, there is no additional software required.

Figure 15-7. High-Speed Data Format



Note: High-Speed mode is 3.4 Mbps, provided correct system clock frequency is set and there is appropriate pull strength on SCL and SDA lines.

15.3.3 Interrupts

The I^{2}C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master arbitration lost
- Master transaction error
- Master bus timeout
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

The I^{2}C master and I^{2}C slave modules have separate interrupt signals. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

15.3.3.1 I^{2}C Master Interrupts

The I^{2}C master module generates an interrupt when a transaction completes (either transmit or receive), when arbitration is lost, or when an error occurs during a transaction. To enable the I^{2}C master interrupt, software must set the IM bit in the **I^{2}C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition is met, software must check the ERROR and ARBLST bits in the **I^{2}C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction and to ensure that arbitration has not been lost. An error condition is asserted if the last transaction wasn't acknowledged by the slave. If an error is not detected and the master has not lost arbitration, the application can proceed with the transfer. The interrupt is cleared by writing a 1 to the IC bit in the **I^{2}C Master Interrupt Clear (I2CMICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I^{2}C Master Raw Interrupt Status (I2CMRIS)** register.

15.3.3.2 I^{2}C 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 **I^{2}C Slave Interrupt Mask (I2CSIMR)** register. Software determines whether the module should write (transmit) or read (receive) data from the **I^{2}C Slave Data (I2CSDR)** register, by checking the RREQ and TREQ bits of the **I^{2}C Slave Control/Status (I2CSCSR)** register. If the slave module is in receive mode and the first byte of a transfer is received, the FBR bit is set along with the RREQ bit. The interrupt is cleared by setting the DATAIC bit in the **I^{2}C 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 **I^{2}C Slave Interrupt Mask (I2CSIMR)** register and cleared by writing a 1 to the STOPIC and STARTIC bits of the **I^{2}C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I^{2}C Slave Raw Interrupt Status (I2CSRIS)** register.

15.3.4 Loopback Operation

The I^{2}C modules can be placed into an internal loopback mode for diagnostic or debug work by setting the LPBK bit in the **I^{2}C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and are tied to the SDA and SCL signals of the slave module to allow internal testing of the device without having to go through I/O.

15.3.5 Command Sequence Flow Charts

This section details the steps required to perform the various I^{2}C transfer types in both master and slave mode.

15.3.5.1 I^{2}C Master Command Sequences

The figures that follow show the command sequences available for the I^{2}C master.

Figure 15-8. Master Single TRANSMIT

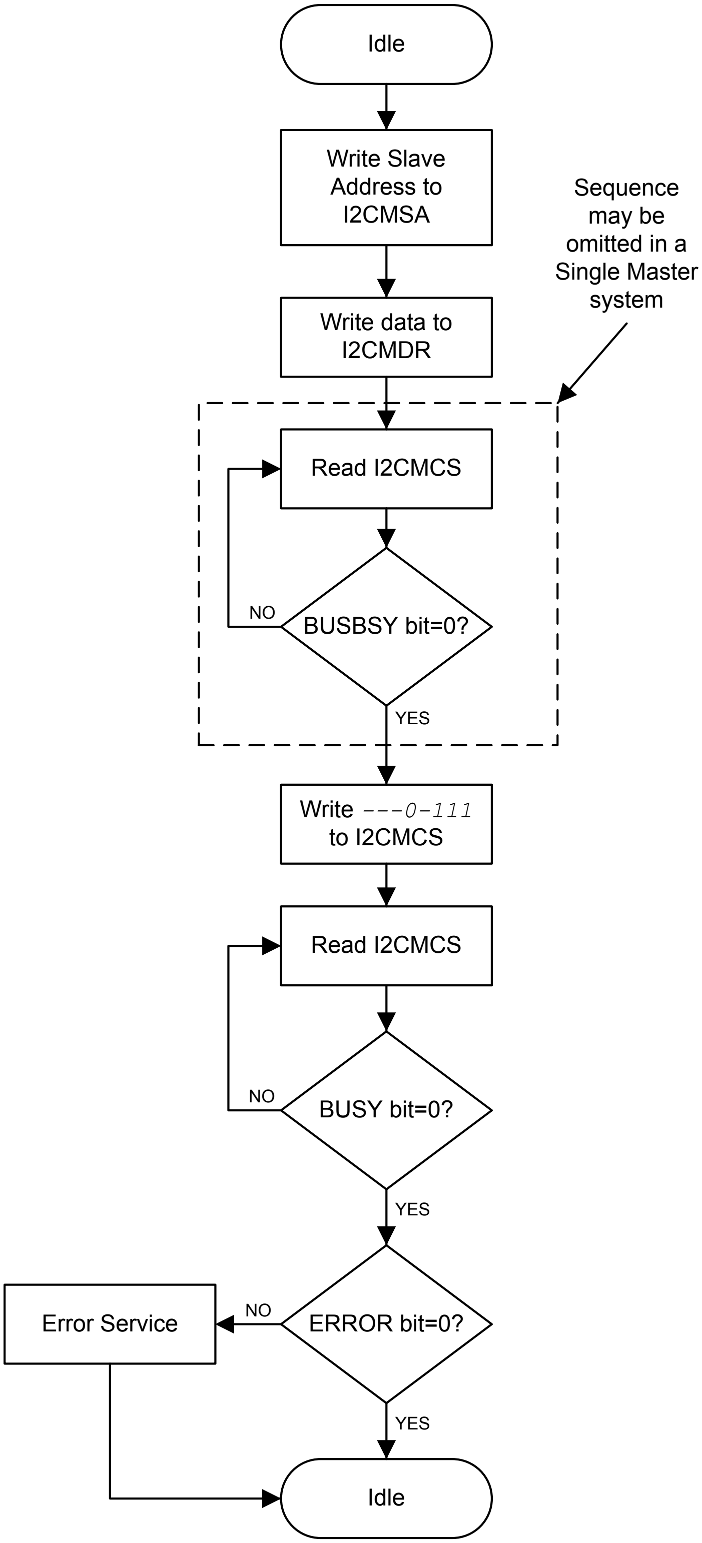


Figure 15-9. Master Single RECEIVE

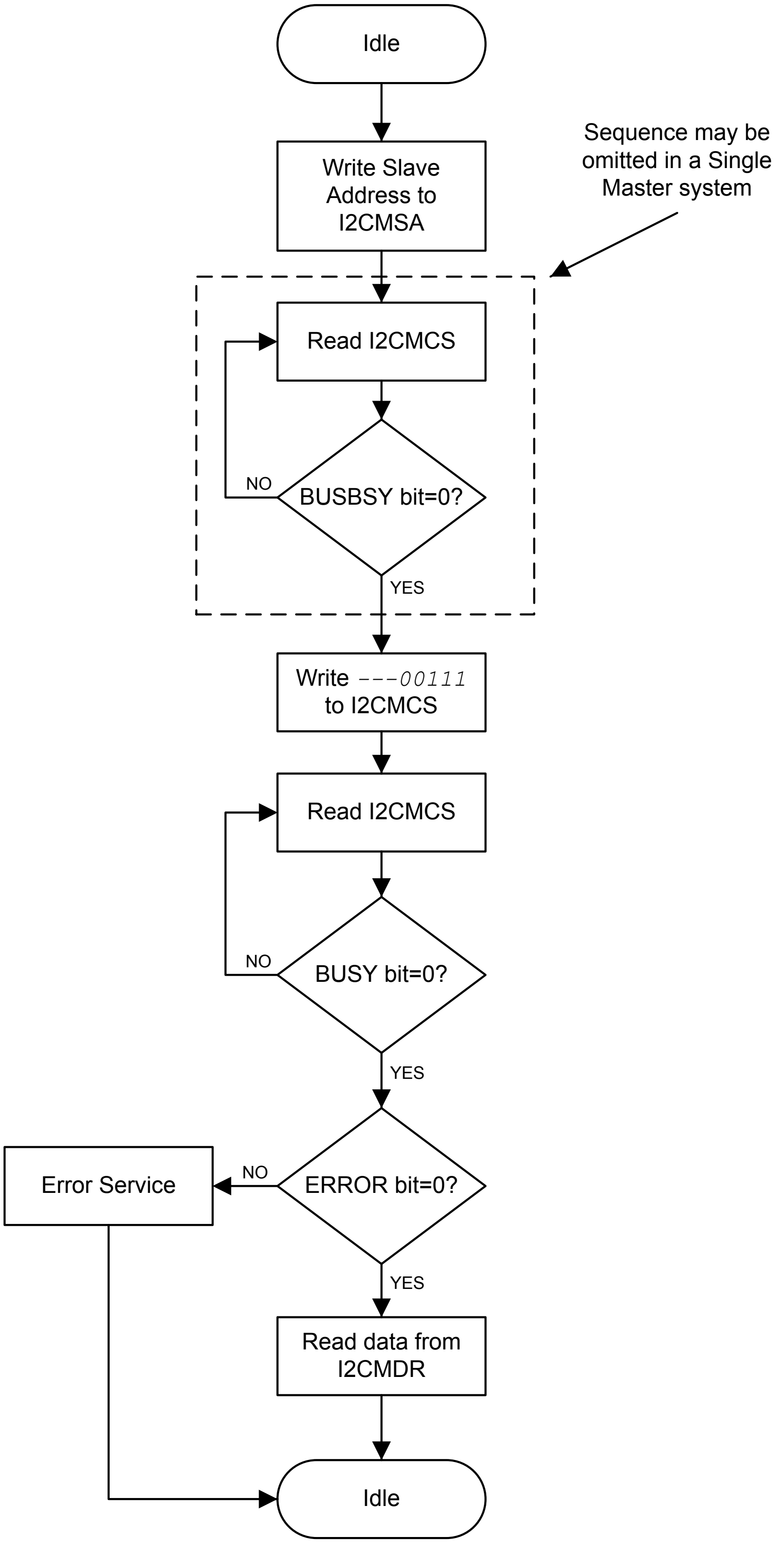


Figure 15-10. Master TRANSMIT of Multiple Data Bytes

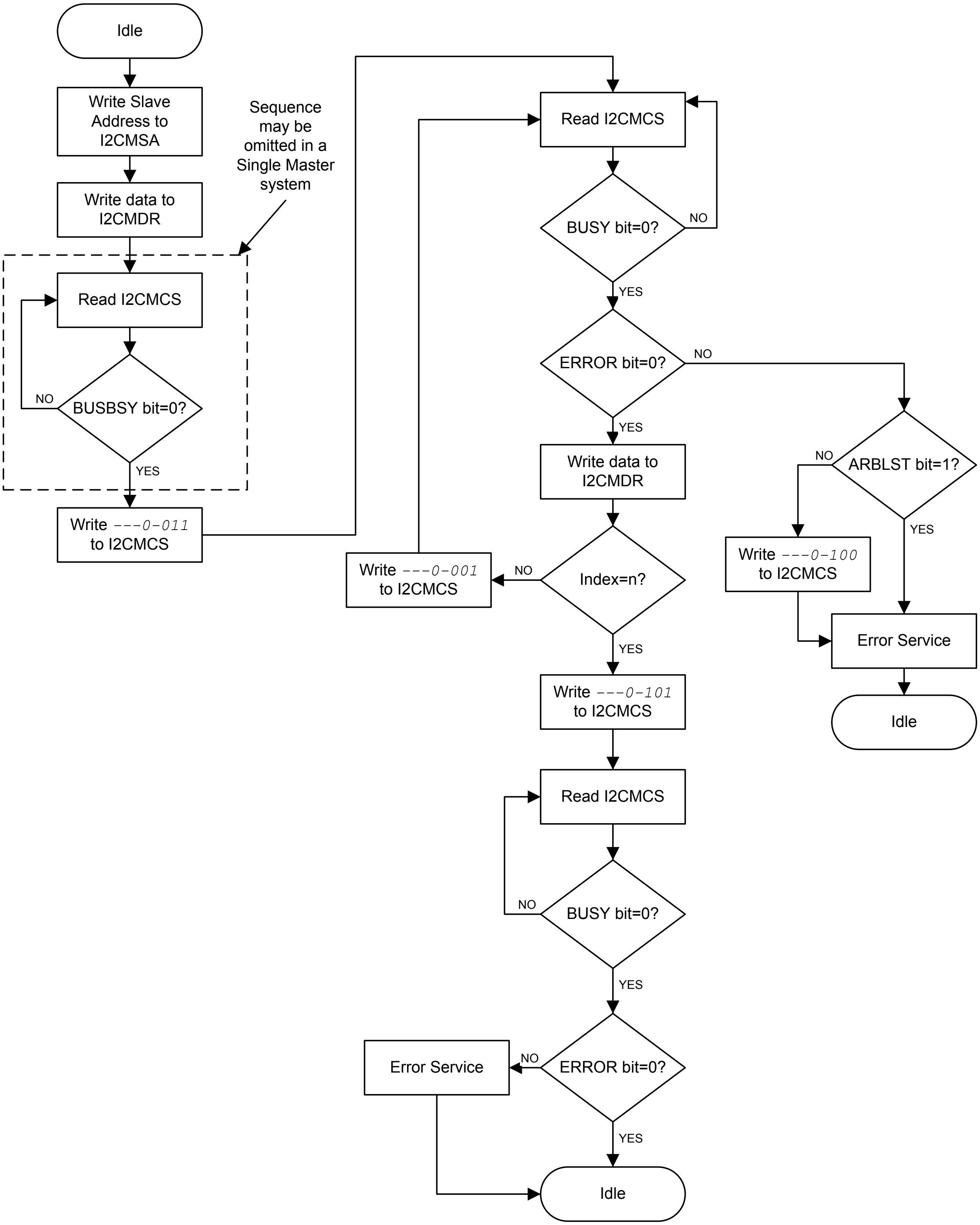


Figure 15-11. Master RECEIVE of Multiple Data Bytes

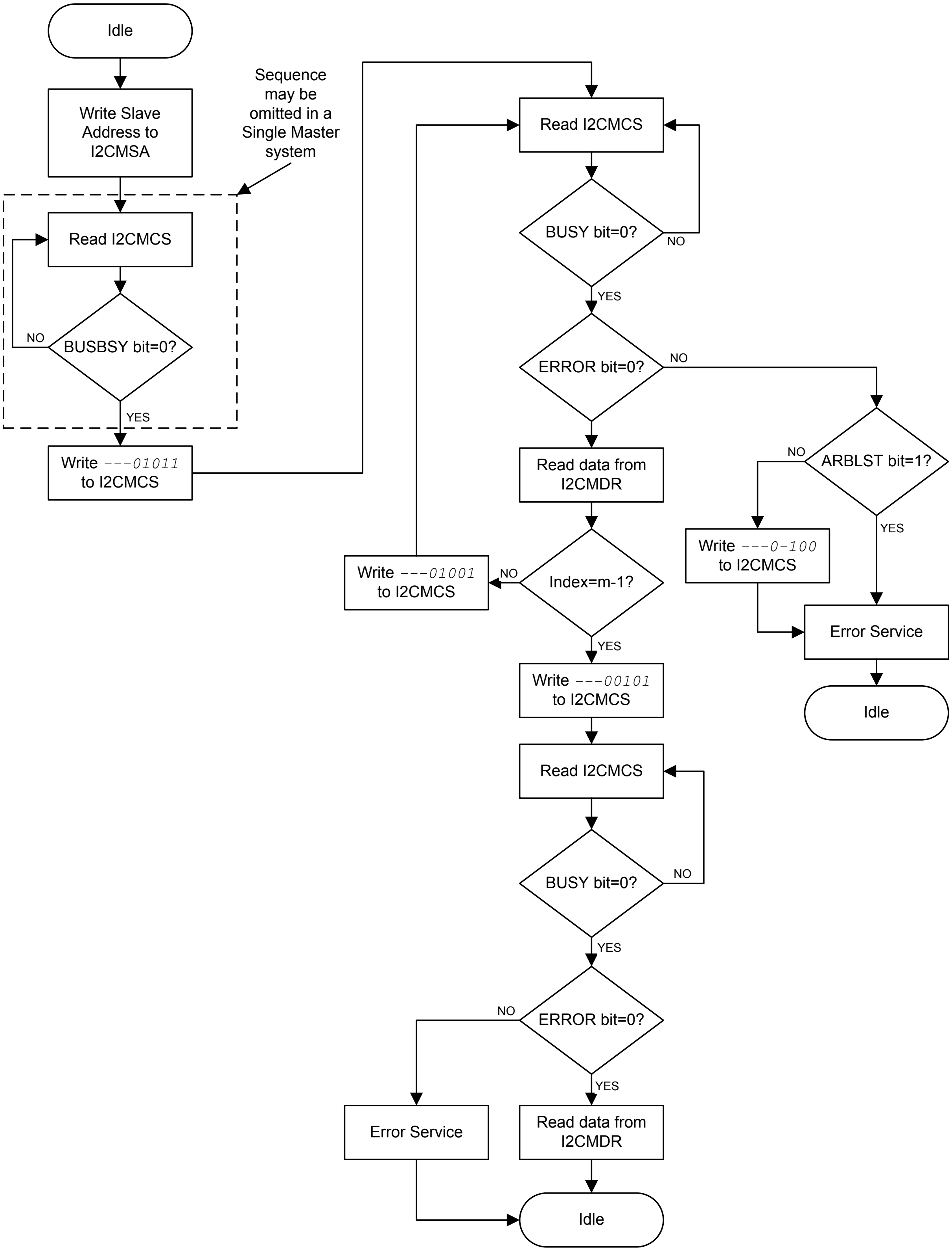


Figure 15-12. Master RECEIVE with Repeated START after Master TRANSMIT

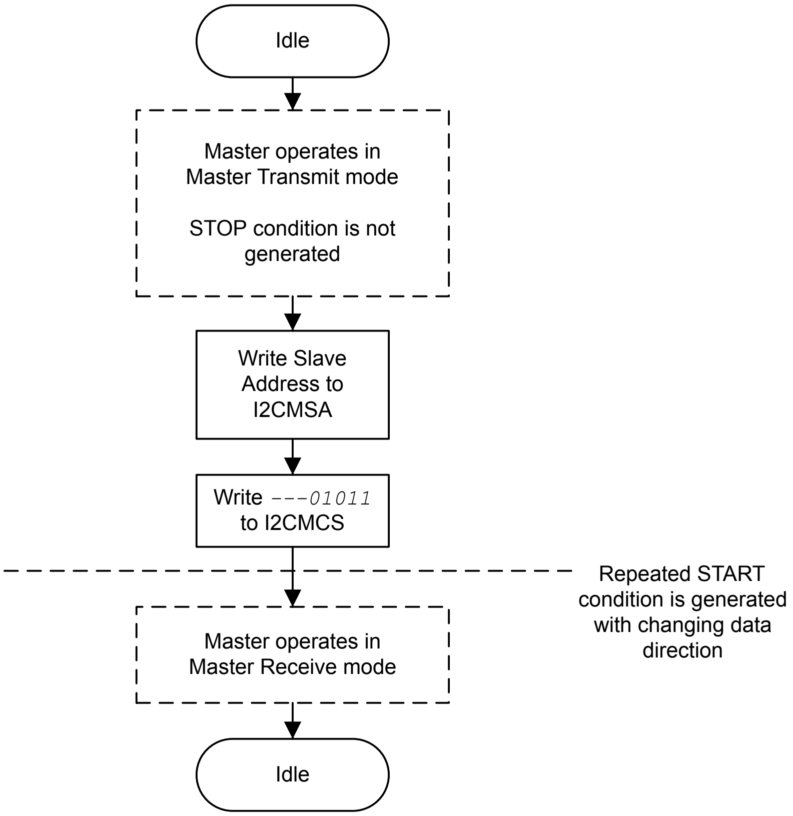

Idle
Master operates in Master Transmit mode
STOP condition is not generated
Write Slave Address to I2CMSA
Write ---01011 to I2CMCS
Repeated START condition is generated with changing data direction
Master operates in Master Receive mode
Idle

Figure 15-13. Master TRANSMIT with Repeated START after Master RECEIVE

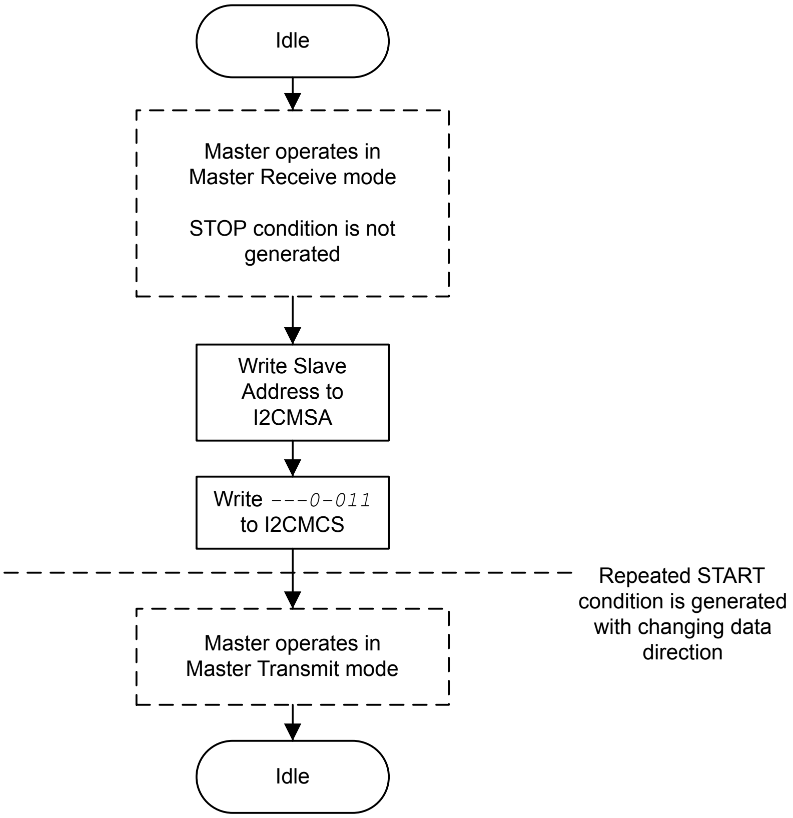
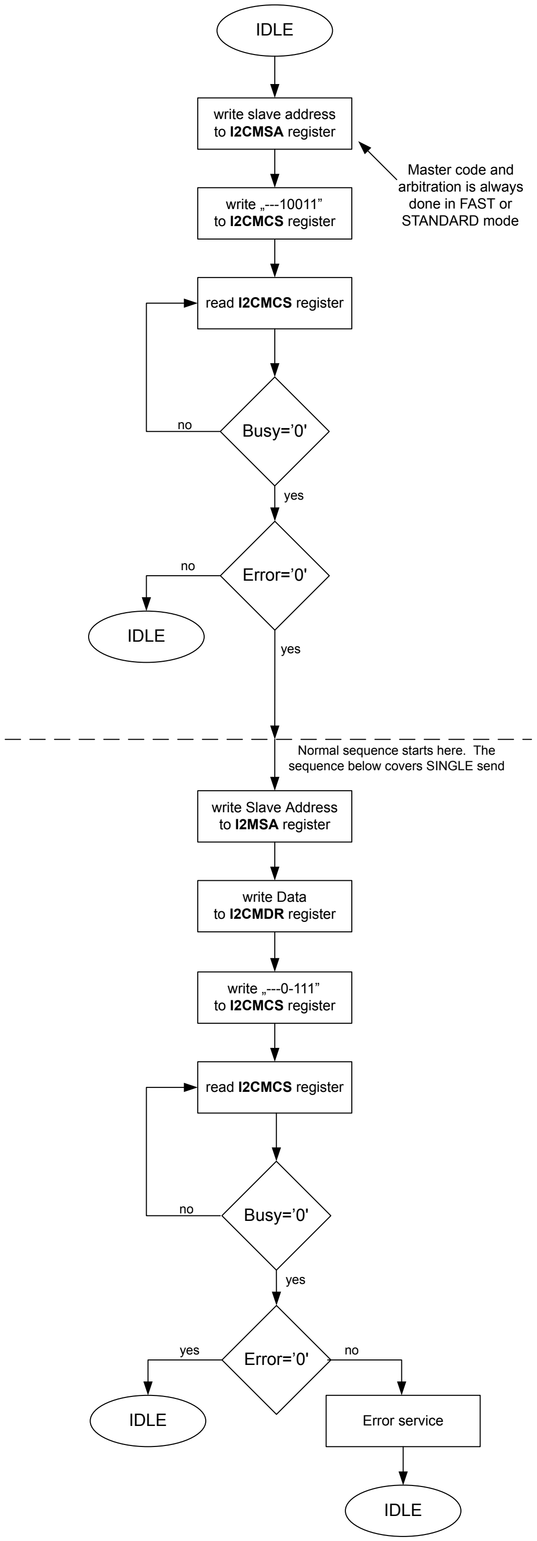

Idle
Master operates in Master Receive mode
STOP condition is not generated
Write Slave Address to I2CMSA
Write ---0-011 to I2CMCS
Repeated START condition is generated with changing data direction
Master operates in Master Transmit mode
Idle

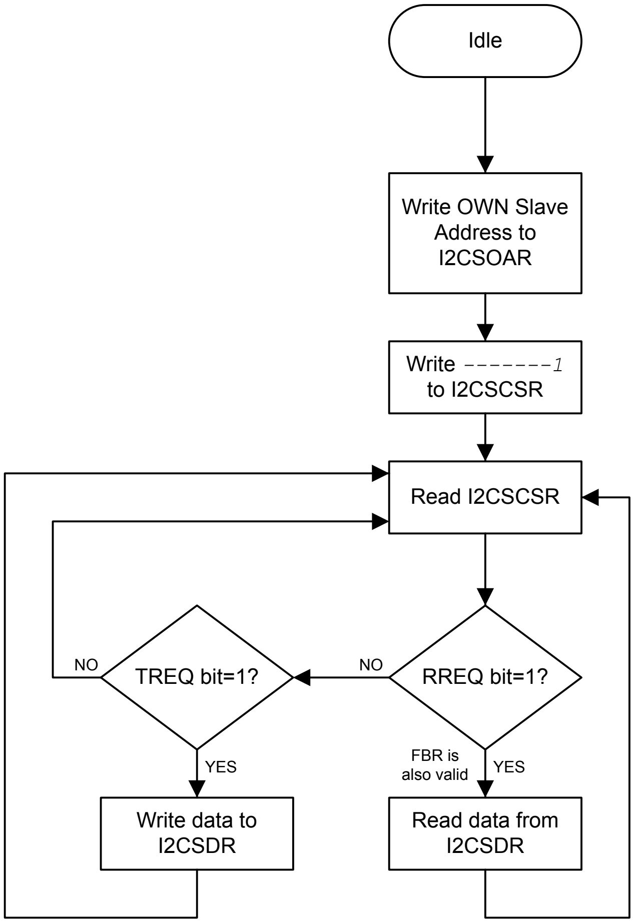
Figure 15-14. Standard High Speed Mode Master Transmit


IDLE
write slave address to I2CMSA register
Master code and arbitration is always done in FAST or STANDARD mode
write „---10011” to I2CMCS register
read I2CMCS register
Busy='0'
no
yes
Error='0'
no
IDLE
yes
Normal sequence starts here. The sequence below covers SINGLE send
write Slave Address to I2MSA register
write Data to I2CMDR register
write „---0-111” to I2CMCS register
read I2CMCS register
Busy='0'
no
yes
Error='0'
yes
no
IDLE
Error service
IDLE

15.3.5.2 I²C Slave Command Sequences

Figure 15-15 on page 944 presents the command sequence available for the I²C slave.

Figure 15-15. Slave Command Sequence



15.4 Initialization and Configuration

15.4.1 Configure the I²C Module to Transmit a Single Byte as a Master

The following example shows how to configure the I²C module to transmit a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I²C clock using the **RCGCI2C** register in the System Control module (see page 326).

2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register in the System Control module (see page 319). To find out which GPIO port to enable, refer to Table 20-5 on page 1115.

3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 603). To determine which GPIOs to configure, see Table 20-4 on page 1111.

4. Enable the I2CSDA pin for open-drain operation. See page 608.

5. Configure the PMCn fields in the **GPIOPCTL** register to assign the I^{2}C signals to the appropriate pins. See page 620 and Table 20-5 on page 1115.

6. Initialize the I^{2}C Master by writing the **I2CMCR** register with a value of 0x0000.0010.

7. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

   ```
   TPR = (System Clock/(2*(SCL_LP + SCL_HP)*SCL_CLK))-1;
   TPR = (20MHz/(2*(6+4)*100000))-1;
   TPR = 9
   ```

   Write the **I2CMTPR** register with the value of 0x0000.0009.

8. Specify the slave address of the master and that the next operation is a Transmit by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.

9. Place data (byte) to be transmitted in the data register by writing the **I2CMDR** register with the desired data.

10. Initiate a single byte transmit of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).

11. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.

12. Check the ERROR bit in the **I2CMCS** register to confirm the transmit was acknowledged.

15.4.2 Configure the I^{2}C Master to High Speed Mode

To configure the I^{2}C master to High Speed mode:

1. Enable the I^{2}C clock using the **RCGCI2C** register in the System Control module (see page 326).

2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register in the System Control module (see page 319). To find out which GPIO port to enable, refer to Table 20-5 on page 1115.

3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 603). To determine which GPIOs to configure, see Table 20-4 on page 1111.

4. Enable the I2CSDA pin for open-drain operation. See page 608.

5. Configure the PMCn fields in the **GPIOPCTL** register to assign the I^{2}C signals to the appropriate pins. See page 620 and Table 20-5 on page 1115.

6. Initialize the I^{2}C Master by writing the **I2CMCR** register with a value of 0x0000.0010.

7. Set the desired SCL clock speed of 3.33 Mbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

```
TPR = (System Clock/(2*(SCL_LP + SCL_HP)*SCL_CLK))-1;
TPR = (80 MHz/(2*(2+1)*3330000))-1;
TPR = 3
```

Write the **I2CMTPR** register with the value of 0x0000.0003.

8. To send the master code byte, software should place the value of the master code byte into the **I2CMSA** register and write the **I2CMCS** register with a value of 0x13.
9. This places the I2C master peripheral in High-speed mode, and all subsequent transfers (until STOP) are carried out at High-speed data rate using the normal **I2CMCS** command bits, without setting the HS bit in the **I2CMCS** register.
10. The transaction is ended by setting the STOP bit in the **I2CMCS** register.
11. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.
12. Check the ERROR bit in the **I2CMCS** register to confirm the transmit was acknowledged.

15.5 Register Map

Table 15-4 on page 946 lists the I²C registers. All addresses given are relative to the I²C base address:

- I²C 0: 0x4002.0000
- I²C 1: 0x4002.1000
- I²C 2: 0x4002.2000
- I²C 3: 0x4002.3000
- I²C 4: 0x400C.0000
- I²C 5: 0x400C.1000

Note that the I²C module clock must be enabled before the registers can be programmed (see page 326). There must be a delay of 3 system clocks after the I²C module clock is enabled before any I²C module registers are accessed.

The hw_i2c.h file in the TivaWare™ Driver Library uses a base address of 0x800 for the I²C slave registers. Be aware when using registers with offsets between 0x800 and 0x818 that TivaWare™ for C Series uses an offset between 0x000 and 0x018 with the slave base address.

Table 15-4. Inter-Integrated Circuit (I²C) Interface Register Map

Offset	Name	Type	Reset	Description	See page
I²C Master					
0x000	I2CMSA	RW	0x0000.0000	I2C Master Slave Address	948
0x004	I2CMCS	RW	0x0000.0020	I2C Master Control/Status	949
0x008	I2CMDR	RW	0x0000.0000	I2C Master Data	954
0x00C	I2CMTPR	RW	0x0000.0001	I2C Master Timer Period	955
0x010	I2CMIMR	RW	0x0000.0000	I2C Master Interrupt Mask	956
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	957

Table 15-4. Inter-Integrated Circuit (I^{2}C) Interface Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	958
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	959
0x020	I2CMCR	RW	0x0000.0000	I2C Master Configuration	960
0x024	I2CMCLKOCNT	RW	0x0000.0000	I2C Master Clock Low Timeout Count	962
0x02C	I2CMBMON	RO	0x0000.0003	I2C Master Bus Monitor	963
0x038	I2CMCR2	RW	0x0000.0000	I2C Master Configuration 2	964
I^{2}C Slave					
0x800	I2CSOAR	RW	0x0000.0000	I2C Slave Own Address	965
0x804	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	966
0x808	I2CSDR	RW	0x0000.0000	I2C Slave Data	968
0x80C	I2CSIMR	RW	0x0000.0000	I2C Slave Interrupt Mask	969
0x810	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	970
0x814	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	971
0x818	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	972
0x81C	I2CSOAR2	RW	0x0000.0000	I2C Slave Own Address 2	973
0x820	I2CSACKCTL	RW	0x0000.0000	I2C Slave ACK Control	974
I^{2}C Status and Control					
0xFC0	I2CPP	RO	0x0000.0001	I2C Peripheral Properties	975
0xFC4	I2CPC	RO	0x0000.0001	I2C Peripheral Configuration	976

15.6 Register Descriptions (I^{2}C Master)

The remainder of this section lists and describes the I^{2}C master registers, in numerical order by address offset.

Register 1: I²C Master Slave Address (I2CMSA), offset 0x000

This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Transmit (Low).

I2C Master Slave Address (I2CMSA)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x000
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								SA							R/S
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:1	SA	RW	0x00	I²C Slave Address This field specifies bits A6 through A0 of the slave address.
0	R/S	RW	0	Receive/Send The R/S bit specifies if the next master operation is a Receive (High) or Transmit (Low). Value Description 0 Transmit 1 Receive

Register 2: I^{2}C Master Control/Status (I2CMCS), offset 0x004

This register accesses status bits when read and control bits when written. When read, the status register indicates the state of the I^{2}C bus controller. When written, the control register configures the I^{2}C controller operation.

The START bit generates the START or REPEATED START condition. The STOP bit determines if the cycle stops at the end of the data cycle or continues to the next transfer cycle, which could be a repeated START. To generate a single transmit cycle, the **I^{2}C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is cleared, and this register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due an error), an interrupt becomes active and the data may be read from the **I2CMDR** register. When the I^{2}C module operates in Master receiver mode, the ACK bit is normally set, causing the I^{2}C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I^{2}C bus controller requires no further data to be transmitted from the slave transmitter.

Read-Only Status Register

I2C Master Control/Status (I2CMCS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x004
Type RO, reset 0x0000.0020

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CLKTO	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	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	CLKTO	RO	0	Clock Timeout Error Value Description 0 No clock timeout error. 1 The clock timeout error has occurred. This bit is cleared when the master sends a STOP condition or if the I^{2}C master is reset.

Bit/Field	Name	Type	Reset	Description
6	BUSBSY	RO	0	Bus Busy Value Description 0 The I^{2}C bus is idle. 1 The I^{2}C bus is busy. The bit changes based on the START and STOP conditions.
5	IDLE	RO	1	I^{2}C Idle Value Description 0 The I^{2}C controller is not idle. 1 The I^{2}C controller is idle.
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	DATACK	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 or the transmit data not being acknowledged.
0	BUSY	RO	0	I^{2}C Busy Value Description 0 The controller is idle. 1 The controller is busy. When the BUSY bit is set, the other status bits are not valid.

Write-Only Control Register

I2C Master Control/Status (I2CMCS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x004
Type WO, reset 0x0000.0020

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											HS	ACK	STOP	START	RUN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:5	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	HS	WO	0	High-Speed Enable Value Description 0 The master operates in Standard, Fast mode, or Fast mode plus as selected by using a value in the **I2CMTPR** register that results in an SCL frequency of 100 kbps for Standard mode, 400 kbps for Fast mode, or 1 Mpbs for Fast mode plus. 1 The master operates in High-Speed mode with transmission speeds up to 3.33 Mbps.
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 15-5 on page 952.
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 15-5 on page 952.

Bit/Field	Name	Type	Reset	Description
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 15-5 on page 952.
0	RUN	WO	0	I^2C Master Enable Value Description 0 This encoding means the master is unable to transmit or receive data. 1 The master is able to transmit or receive data. See field decoding in Table 15-5 on page 952.

Table 15-5. Write Field Decoding for I2CMCS[3:0] Field

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Idle	0	X[a]	0	1	1	START condition followed by TRANSMIT (master goes to the Master Transmit state).
	0	X	1	1	1	START condition followed by a TRANSMIT and STOP condition (master remains in Idle state).
	1	0	0	1	1	START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state).
	1	0	1	1	1	START condition followed by RECEIVE and STOP condition (master remains in Idle state).
	1	1	0	1	1	START condition followed by RECEIVE (master goes to the Master Receive state).
	1	1	1	1	1	Illegal
	All other combinations not listed are non-operations.					NOP

Table 15-5. Write Field Decoding for I2CMCS[3:0] Field *(continued)*

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Master Transmit	X	X	0	0	1	TRANSMIT operation (master remains in Master Transmit state).
	X	X	1	0	0	STOP condition (master goes to Idle state).
	X	X	1	0	1	TRANSMIT followed by STOP condition (master goes to Idle state).
	0	X	0	1	1	Repeated START condition followed by a TRANSMIT (master remains in Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state).
	1	0	0	1	1	Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive state).
	1	0	1	1	1	Repeated START condition followed by a TRANSMIT and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master goes to Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					NOP.
Master Receive	X	0	0	0	1	RECEIVE operation with negative ACK (master remains in Master Receive state).
	X	X	1	0	0	STOP condition (master goes to Idle state).[b]
	X	0	1	0	1	RECEIVE followed by STOP condition (master goes to Idle state).
	X	1	0	0	1	RECEIVE operation (master remains in Master Receive state).
	X	1	1	0	1	Illegal.
	1	0	0	1	1	Repeated START condition followed by RECEIVE operation with a negative ACK (master remains in Master Receive state).
	1	0	1	1	1	Repeated START condition followed by RECEIVE and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master remains in Master Receive state).
	0	X	0	1	1	Repeated START condition followed by TRANSMIT (master goes to Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state).
	All other combinations not listed are non-operations.					NOP.

a. An X in a table cell indicates the bit can be 0 or 1.

b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

Register 3: I^2C Master Data (I2CMDR), offset 0x008

Important: This register is read-sensitive. See the register description for details.

This register contains the data to be transmitted when in the Master Transmit state and the data received when in the Master Receive state.

I2C Master Data (I2CMDR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x008
Type RW, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	RW	0x00	This byte contains the data transferred during a transaction.

Register 4: I^{2}C Master Timer Period (I2CMTPR), offset 0x00C

This register is programmed to set the timer period for the SCL clock and assign the SCL clock to either standard or high-speed mode.

I2C Master Timer Period (I2CMTPR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x00C
Type RW, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								HS	TPR						
Type	RO	RO	RO	RO	RO	RO	RO	RO	WO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	HS	WO	0x0	High-Speed Enable Value Description 0 The SCL Clock Period set by TPR applies to Standard mode (100 Kbps), Fast-mode (400 Kbps), or Fast-mode plus (1 Mbps). 1 The SCL Clock Period set by TPR applies to High-speed mode (3.33 Mbps).
6:0	TPR	RW	0x1	Timer Period This field is used in the equation to configure SCL_PERIOD: $SCL\_PERIOD = 2\times(1 + TPR)\times(SCL\_LP + SCL\_HP)\times CLK\_PRD$ where: SCL_PRD is the SCL line period (I^{2}C clock). TPR is the Timer Period register value (range of 1 to 127). SCL_LP is the SCL Low period (fixed at 6). SCL_HP is the SCL High period (fixed at 4). CLK_PRD is the system clock period in ns.

Register 5: I^2C Master Interrupt Mask (I2CMIMR), offset 0x010

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Master Interrupt Mask (I2CMIMR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x010
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														CLKIM	IM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	CLKIM	RW	0	Clock Timeout Interrupt Mask Value Description 0 The `CLKRIS` interrupt is suppressed and not sent to the interrupt controller. 1 The clock timeout interrupt is sent to the interrupt controller when the `CLKRIS` bit in the **I2CMRIS** register is set.
0	IM	RW	0	Master Interrupt Mask Value Description 0 The `RIS` interrupt is suppressed and not sent to the interrupt controller. 1 The master interrupt is sent to the interrupt controller when the `RIS` bit in the **I2CMRIS** register is set.

Register 6: I²C Master Raw Interrupt Status (I2CMRIS), offset 0x014

This register specifies whether an interrupt is pending.

I2C Master Raw Interrupt Status (I2CMRIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
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														CLKRIS	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
31:2	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	CLKRIS	RO	0	Clock Timeout Raw Interrupt Status Value Description 0 No interrupt. 1 The clock timeout interrupt is pending. This bit is cleared by writing a 1 to the `CLKIC` bit in the **I2CMICR** register.
0	RIS	RO	0	Master Raw Interrupt Status Value Description 0 No interrupt. 1 A master interrupt is pending. This bit is cleared by writing a 1 to the `IC` bit in the **I2CMICR** register.

Register 7: I^{2}C Master Masked Interrupt Status (I2CMMIS), offset 0x018

This register specifies whether an interrupt was signaled.

I2C Master Masked Interrupt Status (I2CMMIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
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														CLKMIS	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
31:2	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	CLKMIS	RO	0	Clock Timeout Masked Interrupt Status Value Description 0 No interrupt. 1 An unmasked clock timeout interrupt was signaled and is pending. This bit is cleared by writing a 1 to the `CLKIC` bit in the **I2CMICR** register.
0	MIS	RO	0	Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked master interrupt was signaled and is pending. This bit is cleared by writing a 1 to the `IC` bit in the **I2CMICR** register.

Register 8: I^{2}C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw and masked interrupts.

I2C Master Interrupt Clear (I2CMICR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
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														CLKIC	IC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:2	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	CLKIC	WO	0	Clock Timeout Interrupt Clear Writing a 1 to this bit clears the `CLKRIS` bit in the **I2CMRIS** register and the `CLKMIS` bit in the **I2CMMIS** register. A read of this register returns no meaningful data.
0	IC	WO	0	Master Interrupt Clear Writing a 1 to this bit clears the `RIS` bit in the **I2CMRIS** register and the `MIS` bit in the **I2CMMIS** register. A read of this register returns no meaningful data.

Register 9: I^{2}C Master Configuration (I2CMCR), offset 0x020

This register configures the mode (Master or Slave), enables the glitch filter, and sets the interface for test mode loopback.

I2C Master Configuration (I2CMCR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x020
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									GFE	SFE	MFE	reserved			LPBK
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	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	GFE	RW	0	I^{2}C Glitch Filter Enable Value Description 0 I^{2}C glitch filter is disabled. 1 I^{2}C glitch filter is enabled. Use the GFPW bit in the **I^{2}C Master Configuration 2 (I2CMCR2)** register to program the pulse width.
5	SFE	RW	0	I^{2}C Slave Function Enable Value Description 0 Slave mode is disabled. 1 Slave mode is enabled.
4	MFE	RW	0	I^{2}C Master Function Enable Value Description 0 Master mode is disabled. 1 Master mode is enabled.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
0	LPBK	RW	0	I^{2}C Loopback Value Description 0 Normal operation. 1 The controller in a test mode loopback configuration.

Register 10: I^2C Master Clock Low Timeout Count (I2CMCLKOCNT), offset 0x024

This register contains the upper 8 bits of a 12-bit counter that can be used to keep the timeout limit for clock stretching by a remote slave. The lower four bits of the counter are not user visible and are always 0x0.

Note: The Master Clock Low Timeout counter counts for the entire time `SCL` is held Low continuously. If `SCL` is deasserted at any point, the Master Clock Low Timeout Counter is reloaded with the value in the **I2CMCLKOCNT** register and begins counting down from this value.

I2C Master Clock Low Timeout Count (I2CMCLKOCNT)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x024
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CNTL							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CNTL	RW	0	I^2C Master Count This field contains the upper 8 bits of a 12-bit counter for the clock low timeout count. **Note:** The value of `CNTL` must be greater than 0x1.

Register 11: I²C Master Bus Monitor (I2CMBMON), offset 0x02C

This register is used to determine the SCL and SDA signal status.

I2C Master Bus Monitor (I2CMBMON)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x02C
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														SDA	SCL
Type	RO	RO	RO	RO	RO	RO	RO	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
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	SDA	RO	1	I²C SDA Status Value Description 0 The I2CSDA signal is low. 1 The I2CSDA signal is high.
0	SCL	RO	1	I²C SCL Status Value Description 0 The I2CSCL signal is low. 1 The I2CSCL signal is high.

Register 12: I^2C Master Configuration 2 (I2CMCR2), offset 0x038

This register can be programmed to select the pulse width for glitch suppression, measured in system clocks.

I2C Master Configuration 2 (I2CMCR2)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x038
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									GFPW			reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	GFPW	RW	0	I^2C Glitch Filter Pulse Width This field controls the pulse width select for glitch suppression on the SCL and SDA lines. Glitch suppression values can be programmed relative to system clocks. Value Description 0x0 Bypass 0x1 1 clock 0x2 2 clocks 0x3 3 clocks 0x4 4 clocks 0x5 8 clocks 0x6 16 clocks 0x7 31 clocks
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

15.7 Register Descriptions (I^2C Slave)

The remainder of this section lists and describes the I^2C slave registers, in numerical order by address offset.

Register 13: I²C Slave Own Address (I2CSOAR), offset 0x800

This register consists of seven address bits that identify the TM4C1232C3PM I²C device on the I²C bus.

I2C Slave Own Address (I2CSOAR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x800
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									OAR						
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	OAR	RW	0x00	I²C Slave Own Address This field specifies bits A6 through A0 of the slave address.

Register 14: I^2C Slave Control/Status (I2CSCSR), offset 0x804

This register functions as a control register when written, and a status register when read.

Read-Only Status Register

I2C Slave Control/Status (I2CSCSR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x804
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												OAR2SEL	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
31:4	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	OAR2SEL	RO	0	OAR2 Address Matched Value Description 0 Either the address is not matched or the match is in legacy mode. 1 OAR2 address matched and ACKed by the slave. This bit gets reevaluated after every address comparison.
2	FBR	RO	0	First Byte Received Value Description 0 The first byte has not been received. 1 The first byte following the slave's own address has been received. This bit is only valid when the RREQ bit is set and is automatically cleared when data has been read from the **I2CSDR** register. **Note:** This bit is not used for slave transmit operations.

Bit/Field	Name	Type	Reset	Description
1	TREQ	RO	0	Transmit Request Value Description 0 No outstanding transmit request. 1 The I²C controller has been addressed as a slave transmitter and is using clock stretching to delay the master until data has been written to the **I2CSDR** register.
0	RREQ	RO	0	Receive Request Value Description 0 No outstanding receive data. 1 The I²C controller has outstanding receive data from the I²C master and is using clock stretching to delay the master until the data has been read from the **I2CSDR** register.

Write-Only Control Register

I2C Slave Control/Status (I2CSCSR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x804
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															DA
Type	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

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 I²C slave operation. 1 Enables the I²C slave operation. Once this bit has been set, it should not be set again unless it has been cleared by writing a 0 or by a reset, otherwise transfer failures may occur.

Register 15: I^2C Slave Data (I2CSDR), offset 0x808

Important: This register is read-sensitive. See the register description for details.

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

I2C Slave Data (I2CSDR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x808
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	RW	0x00	Data for Transfer This field contains the data for transfer during a slave receive or transmit operation.

Register 16: I²C Slave Interrupt Mask (I2CSIMR), offset 0x80C

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Slave Interrupt Mask (I2CSIMR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x80C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													STOPIM	STARTIM	DATAIM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0	Software should not rely on the value of 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	RW	0	Stop Condition Interrupt Mask Value Description 0 The `STOPRIS` interrupt is suppressed and not sent to the interrupt controller. 1 The STOP condition interrupt is sent to the interrupt controller when the `STOPRIS` bit in the **I2CSRIS** register is set.
1	STARTIM	RW	0	Start Condition Interrupt Mask Value Description 0 The `STARTRIS` interrupt is suppressed and not sent to the interrupt controller. 1 The START condition interrupt is sent to the interrupt controller when the `STARTRIS` bit in the **I2CSRIS** register is set.
0	DATAIM	RW	0	Data Interrupt Mask Value Description 0 The `DATARIS` interrupt is suppressed and not sent to the interrupt controller. 1 The data received or data requested interrupt is sent to the interrupt controller when the `DATARIS` bit in the **I2CSRIS** register is set.

Register 17: I^{2}C Slave Raw Interrupt Status (I2CSRIS), offset 0x810

This register specifies whether an interrupt is pending.

I2C Slave Raw Interrupt Status (I2CSRIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x810
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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPRIS	RO	0	Stop Condition Raw Interrupt Status Value Description 0 No interrupt. 1 A STOP condition interrupt is pending. This bit is cleared by writing a 1 to the `STOPIC` bit in the **I2CSICR** register.
1	STARTRIS	RO	0	Start Condition Raw Interrupt Status Value Description 0 No interrupt. 1 A START condition interrupt is pending. This bit is cleared by writing a 1 to the `STARTIC` bit in the **I2CSICR** register.
0	DATARIS	RO	0	Data Raw Interrupt Status Value Description 0 No interrupt. 1 A data received or data requested interrupt is pending. This bit is cleared by writing a 1 to the `DATAIC` bit in the **I2CSICR** register.

Register 18: I^{2}C Slave Masked Interrupt Status (I2CSMIS), offset 0x814

This register specifies whether an interrupt was signaled.

I2C Slave Masked Interrupt Status (I2CSMIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x814
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	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPMIS	RO	0	Stop Condition Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked STOP condition interrupt was signaled is pending. This bit is cleared by writing a 1 to the `STOPIC` bit in the **I2CSICR** register.
1	STARTMIS	RO	0	Start Condition Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked START condition interrupt was signaled is pending. This bit is cleared by writing a 1 to the `STARTIC` bit in the **I2CSICR** register.
0	DATAMIS	RO	0	Data Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked data received or data requested interrupt was signaled is pending. This bit is cleared by writing a 1 to the `DATAIC` bit in the **I2CSICR** register.

Register 19: I^{2}C Slave Interrupt Clear (I2CSICR), offset 0x818

This register clears the raw interrupt. A read of this register returns no meaningful data.

I2C Slave Interrupt Clear (I2CSICR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x818
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	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
31:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIC	WO	0	Stop Condition Interrupt Clear Writing a 1 to this bit clears the `STOPRIS` bit in the **I2CSRIS** register and the `STOPMIS` 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 `STARTRIS` bit in the **I2CSRIS** register and the `STARTMIS` 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 `STOPRIS` bit in the **I2CSRIS** register and the `STOPMIS` bit in the **I2CSMIS** register. A read of this register returns no meaningful data.

Register 20: I^{2}C Slave Own Address 2 (I2CSOAR2), offset 0x81C

This register consists of seven address bits that identify the alternate address for the I^{2}C device on the I^{2}C bus.

I2C Slave Own Address 2 (I2CSOAR2)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x81C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								OAR2EN	OAR2						
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	OAR2EN	RW	0	I^{2}C Slave Own Address 2 Enable Value Description 0 The alternate address is disabled. 1 Enables the use of the alternate address in the OAR2 field.
6:0	OAR2	RW	0x00	I^{2}C Slave Own Address 2 This field specifies the alternate OAR2 address.

Register 21: I^{2}C Slave ACK Control (I2CSACKCTL), offset 0x820

This register enables the I^{2}C slave to NACK for invalid data or command or ACK for valid data or command. The I^{2}C clock is pulled low after the last data bit until this register is written.

I2C Slave ACK Control (I2CSACKCTL)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0x820
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														ACKOVAL	ACKOEN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	ACKOVAL	RW	0	I^{2}C Slave ACK Override Value Value Description 0 An ACK is sent indicating valid data or command. 1 A NACK is sent indicating invalid data or command.
0	ACKOEN	RW	0	I^{2}C Slave ACK Override Enable Value Description 0 A response in not provided. 1 An ACK or NACK is sent according to the value written to the ACKOVAL bit.

15.8 Register Descriptions (I^{2}C Status and Control)

The remainder of this section lists and describes the I^{2}C status and control registers, in numerical order by address offset.

Register 22: I^{2}C Peripheral Properties (I2CPP), offset 0xFC0

The **I2CPP** register provides information regarding the properties of the I^{2}C module.

I2C Peripheral Properties (I2CPP)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0xFC0
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															HS
Type	RO	RO	RO	RO	RO	RO	RO	RO	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	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	HS	RO	0x1	High-Speed Capable

Value	Description
0	The interface is capable of Standard, Fast, or Fast mode plus operation.
1	The interface is capable of High-Speed operation.

Register 23: I^2C Peripheral Configuration (I2CPC), offset 0xFC4

The **I2CPC** register allows software to enable features present in the I^2C module.

I2C Peripheral Configuration (I2CPC)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
Offset 0xFC4
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															HS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	HS	RW	1	High-Speed Capable Value Description 0 The interface is set to Standard, Fast or Fast mode plus operation. 1 The interface is set to High-Speed operation. Note that this encoding may only be used if the HS bit in the **I2CPP** register is set. Otherwise, this encoding is not available.

16 Controller Area Network (CAN) Module

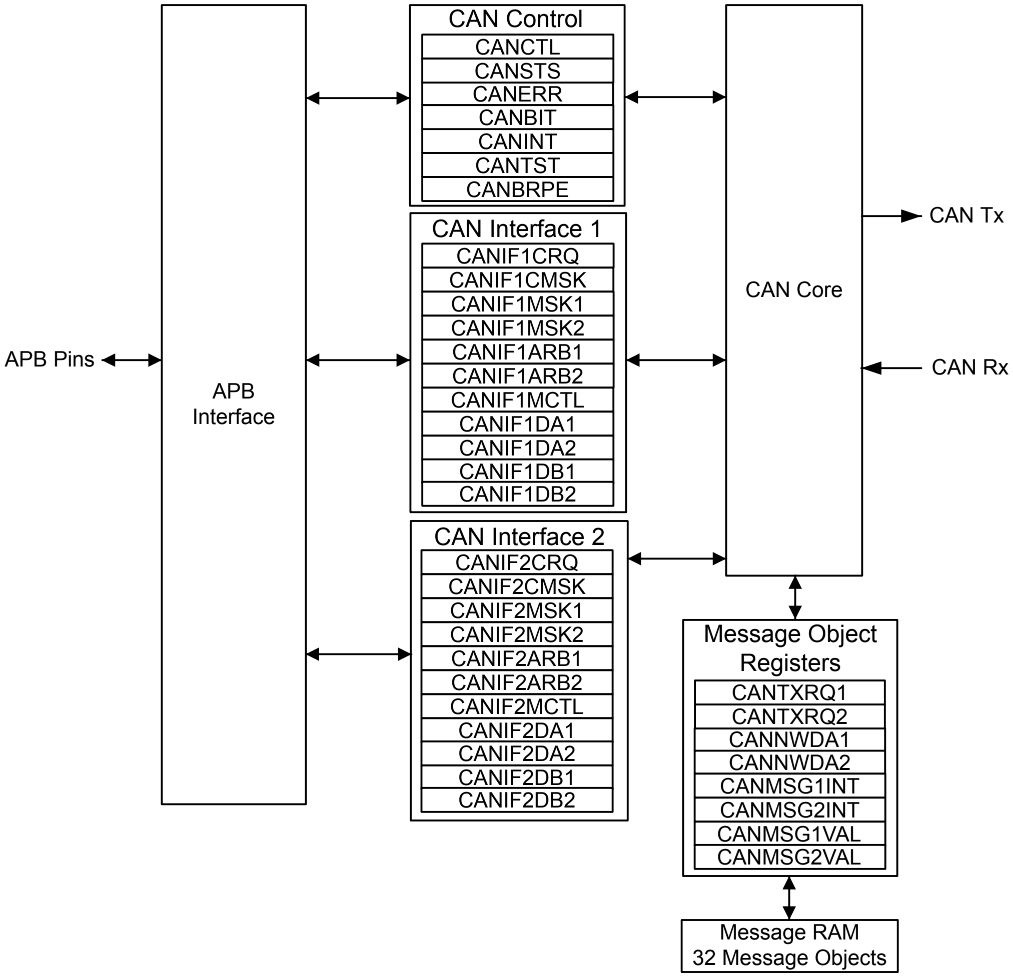
Controller Area Network (CAN) is a multicast, shared serial bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically-noisy environments and can utilize a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally 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 TM4C1232C3PM microcontroller includes one CAN unit 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

16.1 Block Diagram

Figure 16-1. CAN Controller Block Diagram



16.2 Signal Description

The following table lists the external signals of the CAN controller and describes the function of each. The CAN controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the CAN signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 603) should be set to choose the CAN controller function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 620) 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 581.

Table 16-1. Controller Area Network Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
CAN0Rx	28 58 59	PF0 (3) PB4 (8) PE4 (8)	I	TTL	CAN module 0 receive.
CAN0Tx	31 57 60	PF3 (3) PB5 (8) PE5 (8)	O	TTL	CAN module 0 transmit.

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

16.3 Functional Description

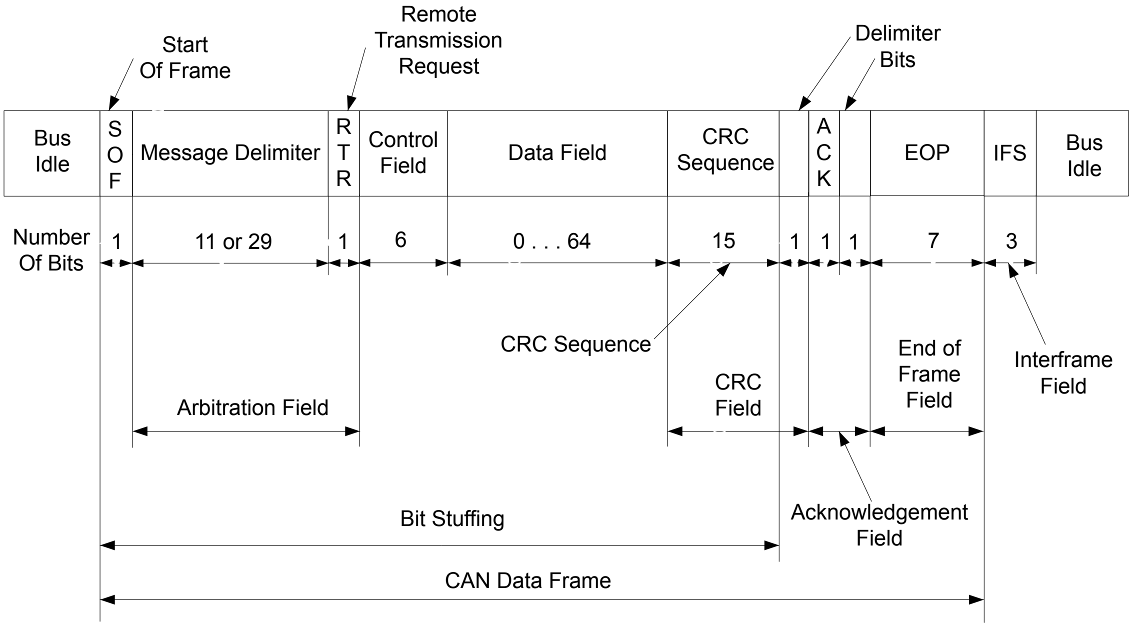
The TM4C1232C3PM CAN controller conforms to the CAN protocol version 2.0 (parts A and B). Message transfers that include data, remote, error, and overload frames with an 11-bit identifier (standard) or a 29-bit identifier (extended) are supported. Transfer rates can be programmed up to 1 Mbps.

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 16-2.

Figure 16-2. 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 TM4C1232C3PM memory map, so the TM4C1232C3PM CAN controller provides an interface to communicate with the message memory via two CAN interface register sets for communicating with the message objects. These two interfaces must be used to read or write to each message object. The two message object interfaces allow parallel access to the CAN controller message objects when multiple objects may have new information that must be processed. In general, one interface is used for transmit data and one for receive data.

16.3.1 Initialization

To use the CAN controller, the peripheral clock must be enabled using the **RCGC0** register (see page 423). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register (see page 429). To find out which GPIO port to enable, refer to Table 20-4 on page 1111. Set the GPIO AFSEL bits for the appropriate pins (see page 603). Configure the PMCn fields in the **GPIOPCTL** register to assign the CAN signals to the appropriate pins. See page 620 and Table 20-5 on page 1115.

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.

16.3.2 Operation

Two sets of CAN Interface Registers (**CANIF1x** and **CANIF2x**) are used to access the message objects in the Message RAM. The CAN controller coordinates transfers to and from the Message RAM to and from the registers. The two sets are independent and identical and can be used to queue transactions. Generally, one interface is used to transmit data and one is used to receive data.

Once the CAN module is initialized and the INIT bit in the **CANCTL** register is cleared, the CAN module synchronizes itself to the CAN bus and starts the message transfer. As each message is received, it goes through the message handler's filtering process, and if it passes through the filter, is stored in the message object specified by the MNUM bit in the **CAN IFn Command Request (CANIFnCRQ)** register. The whole message (including all arbitration bits, data-length code, and eight data bytes) is stored in the message object. If the Identifier Mask (the MSK bits in the **CAN IFn Mask 1** and **CAN IFn Mask 2 (CANIFnMSKn)** registers) is used, the arbitration bits that are masked to "don't care" may be overwritten in the message object.

The CPU may read or write each message at any time via the CAN Interface Registers. The message handler guarantees data consistency in case of concurrent accesses.

The transmission of message objects is under the control of the software that is managing the CAN hardware. Message objects can be used for one-time data transfers or can be permanent message objects used to respond in a more periodic manner. Permanent message objects have all arbitration and control set up, and only the data bytes are updated. At the start of transmission, the appropriate TXRQST bit in the **CAN Transmission Request n (CANTXRQn)** register and the NEWDAT bit in the **CAN New Data n (CANNWDAn)** register are set. If several transmit messages are assigned to the same message object (when the number of message objects is not sufficient), the whole message object has to be configured before the transmission of this message is requested.

The transmission of any number of message objects may be requested at the same time; they are transmitted according to their internal priority, which is based on the message identifier (MNUM) for the message object, with 1 being the highest priority and 32 being the lowest priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data is discarded when a message is updated before its pending transmission has started. Depending on the configuration of the message object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

Transmission can be automatically started by the reception of a matching remote frame. To enable this mode, set the RMTEN bit in the **CAN IFn Message Control (CANIFnMCTL)** register. A matching received remote frame causes the TXRQST bit to be set, and the message object automatically transfers its data or generates an interrupt indicating a remote frame was requested. A remote frame can be strictly a single message identifier, or it can be a range of values specified in the message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are identified as remote frame requests. The UMASK bit in the **CANIFnMCTL** register enables the MSK bits in the **CANIFnMSKn** register to filter which frames are identified as a remote frame request. The MXTD bit in the **CANIFnMSK2** register should be set if a remote frame request is expected to be triggered by 29-bit extended identifiers.

16.3.3 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.

16.3.4 Configuring a Transmit Message Object

The following steps illustrate how to configure a transmit message object.

1. In the **CAN IFn Command Mask (CANIFnCMASK)** register:

   - Set the WRNRD bit to specify a write to the **CANIFnCMASK** register; specify whether to transfer the IDMASK, DIR, and MXTD of the message object into the **CAN IFn** registers using the MASK bit

   - Specify whether to transfer the ID, DIR, XTD, and MSGVAL of the message object into the interface registers using the ARB bit

   - Specify whether to transfer the control bits into the interface registers using the CONTROL bit

   - Specify whether to clear the INTPND bit in the **CANIFnMCTL** register using the CLRINTPND bit

   - Specify whether to clear the NEWDAT bit in the **CANNWDAn** register using the NEWDAT bit

   - Specify which bits to transfer using the DATAA and DATAB bits

2. In the **CANIFnMSK1** register, use the MSK[15:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[15:0] in this register are used for bits [15:0] of the 29-bit message identifier and are not used for an 11-bit identifier. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

3. In the **CANIFnMSK2** register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

4. For a 29-bit identifier, configure ID[15:0] in the **CANIFnARB1** register for bits [15:0] of the message identifier and ID[12:0] in the **CANIFnARB2** register for bits [28:16] of the message identifier. Set the XTD bit to indicate an extended identifier; set the DIR bit to indicate transmit; and set the MSGVAL bit to indicate that the message object is valid.

5. For an 11-bit identifier, disregard the **CANIFnARB1** register and configure ID[12:2] in the **CANIFnARB2** register for bits [10:0] of the message identifier. Clear the XTD bit to indicate a standard identifier; set the DIR bit to indicate transmit; and set the MSGVAL bit to indicate that the message object is valid.

6. In the **CANIFnMCTL** register:

- Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
- Optionally set the TXIE bit to enable the INTPND bit to be set after a successful transmission
- Optionally set the RMTEN bit to enable the TXRQST bit to be set on the reception of a matching remote frame allowing automatic transmission
- Set the EOB bit for a single message object
- Configure the DLC[3:0] field to specify the size of the data frame. Take care during this configuration not to set the NEWDAT, MSGLST, INTPND or TXRQST bits.

7. Load the data to be transmitted into the **CAN IFn Data (CANIFnDA1, CANIFnDA2, CANIFnDB1, CANIFnDB2)** registers. Byte 0 of the CAN data frame is stored in DATA[7:0] in the **CANIFnDA1** register.
8. Program the number of the message object to be transmitted in the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register.
9. When everything is properly configured, set the TXRQST bit in the **CANIFnMCTL** register. Once this bit is set, the message object is available to be transmitted, depending on priority and bus availability. Note that setting the RMTEN bit in the **CANIFnMCTL** register can also start message transmission if a matching remote frame has been received.

16.3.5 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.

16.3.6 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.

16.3.7 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.

16.3.8 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 16-2. Message Object Configurations

Configuration in CANIFnMCTL	Description
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 1 (set the TXRQST bit of the **CANIFnMCTL** register at reception of the frame to enable transmission) ■ UMASK = 1 or 0	At the reception of a matching remote frame, the TXRQST bit of this message object is set. The rest of the message object remains unchanged, and the controller automatically transfers the data in the message object as soon as possible.
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 0 (do not change the TXRQST bit of the **CANIFnMCTL** register at reception of the frame) ■ UMASK = 0 (ignore mask in the **CANIFnMSKn** register)	At the reception of a matching remote frame, the TXRQST bit of this message object remains unchanged, and the remote frame is ignored. This remote frame is disabled, the data is not transferred and nothing indicates that the remote frame ever happened.
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 0 (do not change the TXRQST bit of the **CANIFnMCTL** register at reception of the frame) ■ UMASK = 1 (use mask (MSK, MXTD, and MDIR in the **CANIFnMSKn** register) for acceptance filtering)	At the reception of a matching remote frame, the TXRQST bit of this message object is cleared. The arbitration and control field (ID + XTD + RMTEN + DLC) from the shift register is stored into the message object in the message RAM, and the NEWDAT bit of this message object is set. The data field of the message object remains unchanged; the remote frame is treated similar to a received data frame. This mode is useful for a remote data request from another CAN device for which the TM4C1232C3PM controller does not have readily available data. The software must fill the data and answer the frame manually.

16.3.9 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.

16.3.10 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 982 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 982 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 982 section to program XTD and ID bits for the message identifier to be received; set the MSGVAL bit to indicate a valid message; and clear the DIR bit to specify receive.

5. In the **CANIFnMCTL** register:

   - Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering

   - Optionally set the RXIE bit to enable the INTPND bit to be set after a successful reception

   - Clear the RMTEN bit to leave the TXRQST bit unchanged

   - Set the EOB bit for a single message object

   - Configure the DLC[3:0] field to specify the size of the data frame

   Take care during this configuration not to set the NEWDAT, MSGLST, INTPND or TXRQST bits.

6. Program the number of the message object to be received in the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register. Reception of the message object begins as soon as a matching frame is available on the CAN bus.

When the message handler stores a data frame in the message object, it stores the received Data Length Code and eight data bytes in the **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** register. Byte 0 of the CAN data frame is stored in DATA[7:0] in the **CANIFnDA1** register. If the Data Length Code is less than 8, the remaining bytes of the message object are overwritten by unspecified values.

The CAN mask registers can be used to allow groups of data frames to be received by a message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are received by a message object. The UMASK bit in the **CANIFnMCTL** register enables the MSK bits in the **CANIFnMSKn** register to filter which frames are received. The MXTD bit in the **CANIFnMSK2** register should be set if only 29-bit extended identifiers are expected by this message object.

16.3.11 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.

16.3.11.1 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 985). 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.

16.3.11.2 Reception of Messages with FIFO Buffers

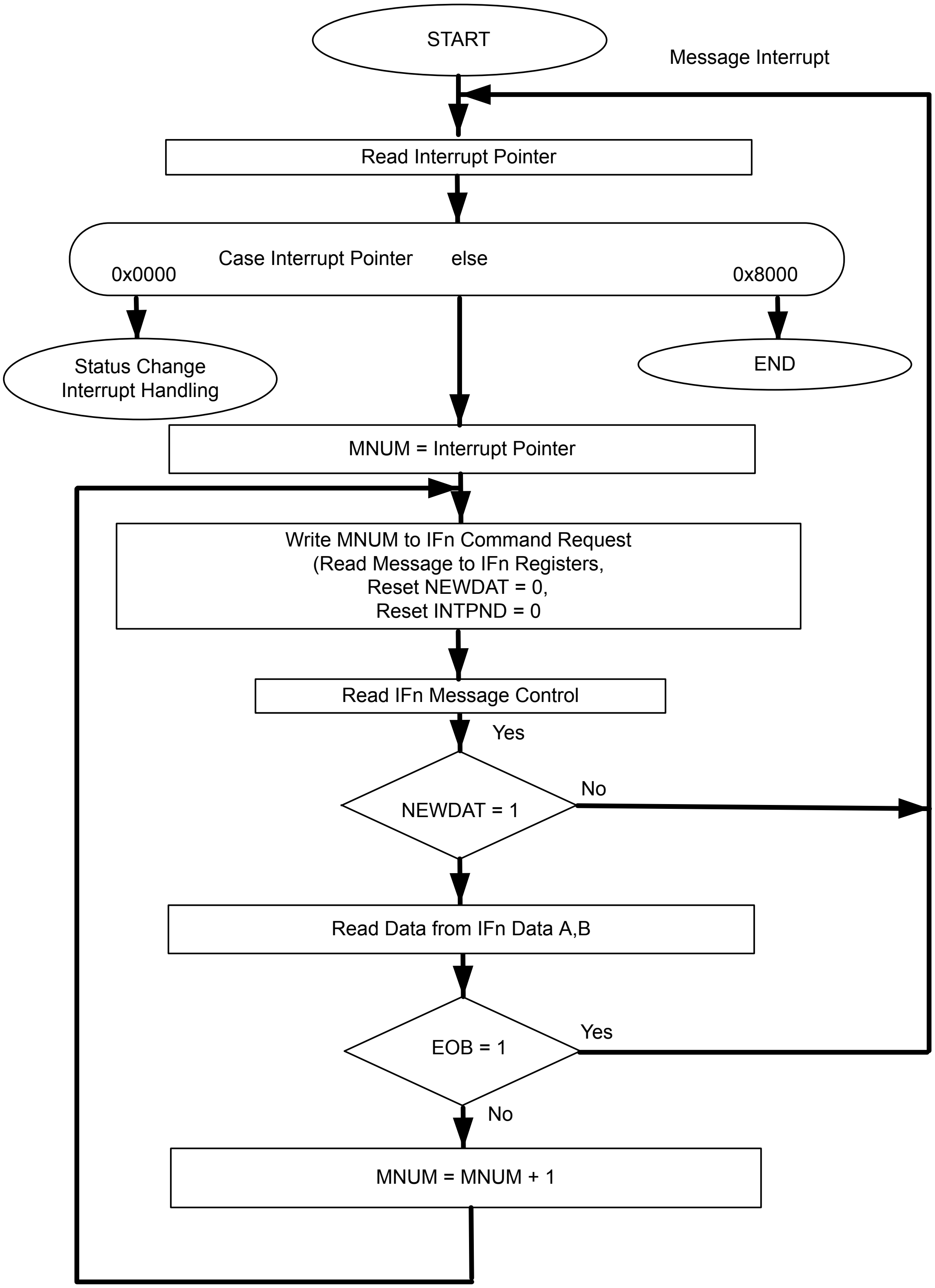
Received messages with identifiers matching to a FIFO buffer are stored starting with the message object with the lowest message number. When a message is stored into a message object of a FIFO buffer, the NEWDAT of the **CANIFnMCTL** register bit of this message object is set. By setting

NEWDAT while EOB is clear, the message object is locked and cannot be written to by the message handler until the CPU has cleared the NEWDAT bit. Messages are stored into a FIFO buffer until the last message object of this FIFO buffer is reached. Until all of the preceding message objects have been released by clearing the NEWDAT bit, all further messages for this FIFO buffer are written into the last message object of the FIFO buffer and therefore overwrite previous messages.

16.3.11.3 Reading from a FIFO Buffer

When the CPU transfers the contents of a message object from a FIFO buffer by writing its number to the **CANIFnCRQ** register, the TXRQST and CLRINTPND bits in the **CANIFnCMSK** register should be set such that the NEWDAT and INTPEND bits in the **CANIFnMCTL** register are cleared after the read. The values of these bits in the **CANIFnMCTL** register always reflect the status of the message object before the bits are cleared. To assure the correct function of a FIFO buffer, the CPU should read out the message objects starting with the message object with the lowest message number. When reading from the FIFO buffer, the user should be aware that a new received message is placed in the message object with the lowest message number for which the NEWDAT bit of the **CANIFnMCTL** register is clear. As a result, the order of the received messages in the FIFO is not guaranteed. Figure 16-3 on page 988 shows how a set of message objects which are concatenated to a FIFO Buffer can be handled by the CPU.

Figure 16-3. Message Objects in a FIFO Buffer



16.3.12 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.

16.3.13 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.

16.3.13.1 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.

16.3.13.2 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.

16.3.13.3 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.

16.3.13.4 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.

16.3.13.5 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.

16.3.14 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.

16.3.15 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 16-4 on page 992): 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 16-3 on page 992). The length of the time quantum (t_q), which is the basic time unit of the bit time, is defined by the CAN controller's input clock (fsys) and the Baud Rate Prescaler (BRP):

t_q = BRP / fsys

The fsys input clock is the system clock frequency as configured by the **RCC** or **RCC2** registers (see page 241 or page 247).

The Synchronization Segment Sync is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync and the Sync is called the phase error of that edge.

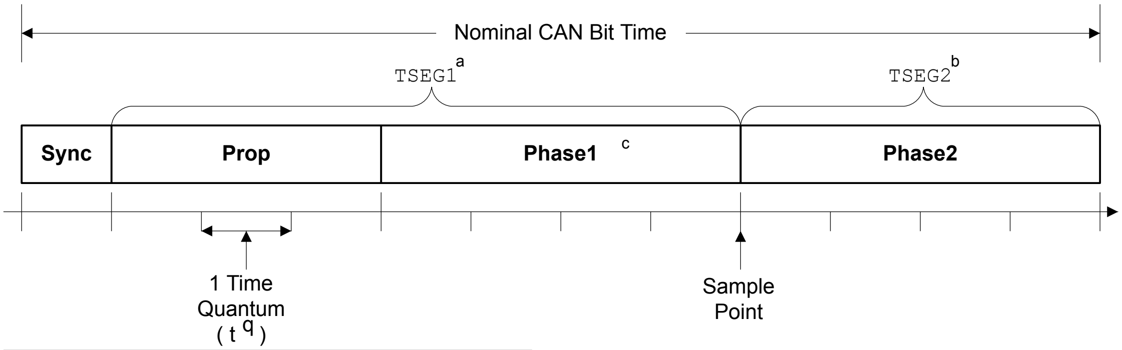
The Propagation Time Segment Prop is intended to compensate for the physical delay times within the CAN network.

The Phase Buffer Segments Phase1 and Phase2 surround the Sample Point.

The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

A given bit rate may be met by different bit-time configurations, but for the proper function of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

Figure 16-4. CAN Bit Time



a. TSEG1 = Prop + Phase1
b. TSEG2 = Phase2
c. Phase1 = Phase2 *or* Phase1 + 1 = Phase2

Table 16-3. CAN Protocol Ranges[a]

Parameter	Range	Remark
BRP	[1 .. 64]	Defines the length of the time quantum t_q. The **CANBRPE** register can be used to extend the range to 1024.
Sync	1 t_q	Fixed length, synchronization of bus input to system clock
Prop	[1 .. 8] t_q	Compensates for the physical delay times
Phase1	[1 .. 8] t_q	May be lengthened temporarily by synchronization
Phase2	[1 .. 8] t_q	May be shortened temporarily by synchronization
SJW	[1 .. 4] t_q	May not be longer than either Phase Buffer Segment

a. This table describes the minimum programmable ranges required by the CAN protocol.

The bit timing configuration is programmed in two register bytes in the **CANBIT** register. In the **CANBIT** register, the four components TSEG2, TSEG1, SJW, and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, for example, SJW (functional range of [1..4]) is represented by only two bits in the SJW bit field. Table 16-4 shows the relationship between the **CANBIT** register values and the parameters.

Table 16-4. CANBIT Register Values

CANBIT Register Field	Setting
TSEG2	Phase2 - 1
TSEG1	Prop + Phase1 - 1
SJW	SJW - 1
BRP	BRP

Therefore, the length of the bit time is (programmed values):

$$[\mathrm{TSEG1} + \mathrm{TSEG2} + 3] \times t_q$$

or (functional values):

$$[\mathrm{Sync} + \mathrm{Prop} + \mathrm{Phase1} + \mathrm{Phase2}] \times t_q$$

The data in the **CANBIT** register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by the BRP field) defines the length of the time quantum, the basic time unit of the bit time; the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the sample point, and occasional synchronizations are controlled by the CAN controller and are evaluated once per time quantum.

The CAN controller translates messages to and from frames. In addition, the controller generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. The bit value is received or transmitted at the sample point. The information processing time (IPT) is the time after the sample point needed to calculate the next bit to be transmitted on the CAN bus. The IPT includes any of the following: retrieving the next data bit, handling a CRC bit, determining if bit stuffing is required, generating an error flag or simply going idle.

The IPT is application-specific but may not be longer than 2 t_q; the CAN's IPT is 0 t_q. Its length is the lower limit of the programmed length of Phase2. In case of synchronization, Phase2 may be shortened to a value less than IPT, which does not affect bus timing.

16.3.16 Calculating the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a required bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the system clock period.

The bit time may consist of 4 to 25 time quanta. Several combinations may lead to the required bit time, allowing iterations of the following steps.

The first part of the bit time to be defined is Prop. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for Prop is converted into time quanta (rounded up to the nearest integer multiple of t_q).

Sync is 1 t_q long (fixed), which leaves (bit time - Prop - 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, that is, Phase2 = Phase1, else Phase2 = Phase1 + 1.

The minimum nominal length of Phase2 has to be regarded as well. Phase2 may not be shorter than the CAN controller's Information Processing Time, which is, depending on the actual implementation, in the range of [0..2] t_q.

The length of the synchronization jump width is set to the least of 4, Phase1 or Phase2.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formula given below:

$$(1 - df) \times fnom \leq fosc \leq (1 + df) \times fnom$$

where:

- df = Maximum tolerance of oscillator frequency
- fosc = Actual oscillator frequency
- fnom = Nominal oscillator frequency

Maximum frequency tolerance must take into account the following formulas:

$$df \leq \frac{(Phase\_seg1, Phase\_seg2)\ \mathrm{min}}{2 \times (13 \times tbit - Phase\_Seg2)}$$

$$df\ \mathrm{max} = 2 \times df \times fnom$$

where:

- Phase1 and Phase2 are from Table 16-3 on page 992
- 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.

16.3.16.1 Example for Bit Timing at High Baud Rate

In this example, the frequency of CAN clock is 25 MHz, and the bit rate is 1 Mbps.

```
bit time = 1 µs = n * tq = 5 * tq
tq = 200 ns
tq = (Baud rate Prescaler)/CAN Clock
Baud rate Prescaler = tq * CAN Clock
Baud rate Prescaler = 200E-9 * 25E6 = 5

tSync = 1 * tq = 200 ns              \\fixed at 1 time quanta

delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
tProp 400 ns = 2 * tq                \\400 is next integer multiple of tq

bit time = tSync + tTSeg1 + tTSeg2 = 5 * tq
bit time = tSync + tProp + tPhase 1 + tPhase2
tPhase 1 + tPhase2 = bit time - tSync - tProp
tPhase 1 + tPhase2 = (5 * tq) - (1 * tq) - (2 * tq)
tPhase 1 + tPhase2 = 2 * tq
tPhase1 = 1 * tq
tPhase2 = 1 * tq                     \\tPhase2 = tPhase1
```

```
tTSeg1 = tProp + tPhase1
tTSeg1 = (2 * $t_q$) + (1 * $t_q$)
tTSeg1 = 3 * $t_q$

tTSeg2 = tPhase2
tTSeg2 = (Information Processing Time + 1) * $t_q$
tTSeg2 = 1 * $t_q$                        \\Assumes IPT=0

tSJW = 1 * $t_q$                         \\Least of 4, Phase1 and Phase2
```

In the above example, the bit field values for the **CANBIT** register are:

TSEG2	= TSeg2 -1 = 1-1 = 0
TSEG1	= TSeg1 -1 = 3-1 = 2
SJW	= SJW -1 = 1-1 = 0
BRP	= Baud rate prescaler - 1 = 5-1 =4

The final value programmed into the **CANBIT** register = 0x0204.

16.3.16.2 Example for Bit Timing at Low Baud Rate

In this example, the frequency of the CAN clock is 50 MHz, and the bit rate is 100 Kbps.

```
bit time = 10 µs = n * $t_q$ = 10 * $t_q$
$t_q$ = 1 µs
$t_q$ = (Baud rate Prescaler)/CAN Clock
Baud rate Prescaler = $t_q$ * CAN Clock
Baud rate Prescaler = 1E-6 * 50E6 = 50

tSync = 1 * $t_q$ = 1 µs                \\fixed at 1 time quanta

delay of bus driver 200 ns
delay of receiver circuit 80 ns
delay of bus line (40m) 220 ns
tProp 1 µs = 1 * $t_q$                  \\1 µs is next integer multiple of $t_q$

bit time = tSync + tTSeg1 + tTSeg2 = 10 * $t_q$
bit time = tSync + tProp + tPhase 1 + tPhase2
tPhase 1 + tPhase2 = bit time - tSync - tProp
tPhase 1 + tPhase2 = (10 * $t_q$) - (1 * $t_q$)  - (1 * $t_q$)
tPhase 1 + tPhase2 = 8 * $t_q$
tPhase1 = 4 * $t_q$
tPhase2 = 4 * $t_q$                     \\tPhase1 = tPhase2
```

```
tTSeg1 = tProp + tPhase1
tTSeg1 = (1 * tq) + (4 * tq)
tTSeg1 = 5 * tq
tTSeg2 = tPhase2
tTSeg2 = (Information Processing Time + 4) × tq
tTSeg2 = 4 * tq                      \\Assumes IPT=0

tSJW = 4 * tq                        \\Least of 4, Phase1, and Phase2
```

TSEG2	= TSeg2 -1 = 4-1 = 3
TSEG1	= TSeg1 -1 = 5-1 = 4
SJW	= SJW -1 = 4-1 = 3
BRP	= Baud rate prescaler - 1 = 50-1 =49

The final value programmed into the **CANBIT** register = 0x34F1.

16.4 Register Map

Table 16-5 on page 996 lists the registers. All addresses given are relative to the CAN base address of:

- CAN0: 0x4004.0000

Note that the CAN controller clock must be enabled before the registers can be programmed (see page 329). There must be a delay of 3 system clocks after the CAN module clock is enabled before any CAN module registers are accessed.

Table 16-5. CAN Register Map

Offset	Name	Type	Reset	Description	See page
0x000	CANCTL	RW	0x0000.0001	CAN Control	998
0x004	CANSTS	RW	0x0000.0000	CAN Status	1000
0x008	CANERR	RO	0x0000.0000	CAN Error Counter	1003
0x00C	CANBIT	RW	0x0000.2301	CAN Bit Timing	1004
0x010	CANINT	RO	0x0000.0000	CAN Interrupt	1005
0x014	CANTST	RW	0x0000.0000	CAN Test	1006
0x018	CANBRPE	RW	0x0000.0000	CAN Baud Rate Prescaler Extension	1008
0x020	CANIF1CRQ	RW	0x0000.0001	CAN IF1 Command Request	1009
0x024	CANIF1CMSK	RW	0x0000.0000	CAN IF1 Command Mask	1010

Table 16-5. CAN Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x028	CANIF1MSK1	RW	0x0000.FFFF	CAN IF1 Mask 1	1013
0x02C	CANIF1MSK2	RW	0x0000.FFFF	CAN IF1 Mask 2	1014
0x030	CANIF1ARB1	RW	0x0000.0000	CAN IF1 Arbitration 1	1016
0x034	CANIF1ARB2	RW	0x0000.0000	CAN IF1 Arbitration 2	1017
0x038	CANIF1MCTL	RW	0x0000.0000	CAN IF1 Message Control	1019
0x03C	CANIF1DA1	RW	0x0000.0000	CAN IF1 Data A1	1022
0x040	CANIF1DA2	RW	0x0000.0000	CAN IF1 Data A2	1022
0x044	CANIF1DB1	RW	0x0000.0000	CAN IF1 Data B1	1022
0x048	CANIF1DB2	RW	0x0000.0000	CAN IF1 Data B2	1022
0x080	CANIF2CRQ	RW	0x0000.0001	CAN IF2 Command Request	1009
0x084	CANIF2CMSK	RW	0x0000.0000	CAN IF2 Command Mask	1010
0x088	CANIF2MSK1	RW	0x0000.FFFF	CAN IF2 Mask 1	1013
0x08C	CANIF2MSK2	RW	0x0000.FFFF	CAN IF2 Mask 2	1014
0x090	CANIF2ARB1	RW	0x0000.0000	CAN IF2 Arbitration 1	1016
0x094	CANIF2ARB2	RW	0x0000.0000	CAN IF2 Arbitration 2	1017
0x098	CANIF2MCTL	RW	0x0000.0000	CAN IF2 Message Control	1019
0x09C	CANIF2DA1	RW	0x0000.0000	CAN IF2 Data A1	1022
0x0A0	CANIF2DA2	RW	0x0000.0000	CAN IF2 Data A2	1022
0x0A4	CANIF2DB1	RW	0x0000.0000	CAN IF2 Data B1	1022
0x0A8	CANIF2DB2	RW	0x0000.0000	CAN IF2 Data B2	1022
0x100	CANTXRQ1	RO	0x0000.0000	CAN Transmission Request 1	1023
0x104	CANTXRQ2	RO	0x0000.0000	CAN Transmission Request 2	1023
0x120	CANNWDA1	RO	0x0000.0000	CAN New Data 1	1024
0x124	CANNWDA2	RO	0x0000.0000	CAN New Data 2	1024
0x140	CANMSG1INT	RO	0x0000.0000	CAN Message 1 Interrupt Pending	1025
0x144	CANMSG2INT	RO	0x0000.0000	CAN Message 2 Interrupt Pending	1025
0x160	CANMSG1VAL	RO	0x0000.0000	CAN Message 1 Valid	1026
0x164	CANMSG2VAL	RO	0x0000.0000	CAN Message 2 Valid	1026

16.5 CAN Register Descriptions

The remainder of this section lists and describes the CAN registers, in numerical order by address offset. There are two sets of Interface Registers that are used to access the Message Objects in the Message RAM: **CANIF1x** and **CANIF2x**. The function of the two sets are identical and are used to queue transactions.

Register 1: CAN Control (CANCTL), offset 0x000

This control register initializes the module and enables test mode and interrupts.

The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or clearing INIT. If the device goes bus-off, it sets INIT, stopping all bus activities. Once INIT has been cleared by the CPU, the device then waits for 129 occurrences of Bus Idle (129 * 11 consecutive High bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters are reset.

During the waiting time after INIT is cleared, each time a sequence of 11 High bits has been monitored, a BITERROR0 code is written to the **CANSTS** register (the LEC field = 0x5), enabling the CPU to readily check whether the CAN bus is stuck Low or continuously disturbed, and to monitor the proceeding of the bus-off recovery sequence.

CAN Control (CANCTL)

CAN0 base: 0x4004.0000
Offset 0x000
Type RW, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								TEST	CCE	DAR	reserved	EIE	SIE	IE	INIT
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TEST	RW	0	Test Mode Enable

Value	Description
0	The CAN controller is operating normally.
1	The CAN controller is in test mode.

Bit/Field	Name	Type	Reset	Description
6	CCE	RW	0	Configuration Change Enable

Value	Description
0	Write accesses to the **CANBIT** register are not allowed.
1	Write accesses to the **CANBIT** register are allowed if the INIT bit is 1.

Bit/Field	Name	Type	Reset	Description
5	DAR	RW	0	Disable Automatic-Retransmission

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	RW	0	Error Interrupt Enable Value / Description 0 — No error status interrupt is generated. 1 — A change in the BOFF or EWARN bits in the **CANSTS** register generates an interrupt.
2	SIE	RW	0	Status Interrupt Enable 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 TXOK, RXOK or LEC bits in the **CANSTS** register generates an interrupt.
1	IE	RW	0	CAN Interrupt Enable Value / Description 0 — Interrupts disabled. 1 — Interrupts enabled.
0	INIT	RW	1	Initialization Value / Description 0 — Normal operation. 1 — Initialization started.

Register 2: CAN Status (CANSTS), offset 0x004

Important: This register is read-sensitive. See the register description for details.

The status register contains information for interrupt servicing such as Bus-Off, error count threshold, and error types.

The LEC field holds the code that indicates the type of the last error to occur on the CAN bus. This field is cleared when a message has been transferred (reception or transmission) without error. The unused error code 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
Offset 0x004
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								BOFF	EWARN	EPASS	RXOK	TXOK	LEC		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	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.

Bit/Field	Name	Type	Reset	Description
6	EWARN	RO	0	Warning Status

Value	Description
0	Both error counters are below the error warning limit of 96.
1	At least one of the error counters has reached the error warning limit of 96.

<table>
<tr><th>Bit/Field</th><th>Name</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>5</td><td>EPASS</td><td>RO</td><td>0</td><td>Error Passive
<table>
<tr><th>Value</th><th>Description</th></tr>
<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>
</table>
</td></tr>
<tr><td>4</td><td>RXOK</td><td>RW</td><td>0</td><td>Received a Message Successfully
<table>
<tr><th>Value</th><th>Description</th></tr>
<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>
</table>
This bit must be cleared by writing a 0 to it.</td></tr>
<tr><td>3</td><td>TXOK</td><td>RW</td><td>0</td><td>Transmitted a Message Successfully
<table>
<tr><th>Value</th><th>Description</th></tr>
<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>
</table>
This bit must be cleared by writing a 0 to it.</td></tr>
</table>

Bit/Field	Name	Type	Reset	Description
2:0	LEC	RW	0x0	Last Error Code This is the type of the last error to occur on the CAN bus.

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	No Event 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
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 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 991 for more information.

CAN Bit Timing (CANBIT)

CAN0 base: 0x4004.0000
Offset 0x00C
Type RW, 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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
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	RW	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 `Phase2` (see Figure 16-4 on page 992). The bit time quanta is defined by the `BRP` field.
11:8	TSEG1	RW	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 `Phase1` (see Figure 16-4 on page 992). The bit time quanta is defined by the `BRP` field.
7:6	SJW	RW	0x0	(Re)Synchronization Jump Width 0x00-0x03: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. During the start of frame (SOF), if the CAN controller detects a phase error (misalignment), it can adjust the length of `TSEG2` or `TSEG1` by the value in `SJW`. So the reset value of 0 adjusts the length by 1 bit time quanta.
5:0	BRP	RW	0x1	Baud Rate Prescaler The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. 0x00-0x03F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. `BRP` defines the number of CAN clock periods that make up 1 bit time quanta, so the reset value is 2 bit time quanta (1+1). The **CANBRPE** register can be used to further divide the bit time.

Register 5: CAN Interrupt (CANINT), offset 0x010

This register indicates the source of the interrupt.

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding the order in which the interrupts occurred. An interrupt remains pending until the CPU has cleared it. If the `INTID` field is not 0x0000 (the default) and the `IE` bit in the **CANCTL** register is set, the interrupt is active. The interrupt line remains active until the `INTID` field is cleared by reading the **CANSTS** register, or until the `IE` bit in the **CANCTL** register is cleared.

Note: Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Interrupt (CANINT)

CAN0 base: 0x4004.0000
Offset 0x010
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INTID															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	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	Reserved
0x8000	Status Interrupt
0x8001-0xFFFF	Reserved

Register 6: CAN Test (CANTST), offset 0x014

This register is used for self-test and external pin access. It is write-enabled by setting the TEST bit in the **CANCTL** register. Different test functions may be combined, however, CAN transfers are affected if the TX bits in this register are not zero.

CAN Test (CANTST)

CAN0 base: 0x4004.0000
Offset 0x014
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								RX	TX		LBACK	SILENT	BASIC	reserved	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a 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

Value	Description
0	The CANnRx pin is low.
1	The CANnRx pin is high.

Bit/Field	Name	Type	Reset	Description
6:5	TX	RW	0x0	Transmit Control Overrides control of the CANnTx pin.

Value	Description
0x0	CAN Module Control CANnTx is controlled by the CAN module; default operation
0x1	Sample Point The sample point is driven on the CANnTx signal. This mode is useful to monitor bit timing.
0x2	Driven Low CANnTx drives a low value. This mode is useful for checking the physical layer of the CAN bus.
0x3	Driven High CANnTx drives a high value. This mode is useful for checking the physical layer of the CAN bus.

<table>
<tr><th>Bit/Field</th><th>Name</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>4</td><td>LBACK</td><td>RW</td><td>0</td><td>Loopback Mode
<table>
<tr><th>Value</th><th>Description</th></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></td></tr>
<tr><td>3</td><td>SILENT</td><td>RW</td><td>0</td><td>Silent Mode
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Silent mode is disabled.</td></tr>
<tr><td>1</td><td>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.</td></tr>
</table></td></tr>
<tr><td>2</td><td>BASIC</td><td>RW</td><td>0</td><td>Basic Mode
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Basic mode is disabled.</td></tr>
<tr><td>1</td><td>Basic mode is enabled. In basic mode, software should use the <b>CANIF1</b> registers as the transmit buffer and use the <b>CANIF2</b> registers as the receive buffer.</td></tr>
</table></td></tr>
<tr><td>1:0</td><td>reserved</td><td>RO</td><td>0x0</td><td>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</td></tr>
</table>

Register 7: CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018

This register is used to further divide the bit time set with the BRP bit in the **CANBIT** register. It is write-enabled by setting the CCE bit in the **CANCTL** register.

CAN Baud Rate Prescaler Extension (CANBRPE)

CAN0 base: 0x4004.0000
Offset 0x018
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												BRPE			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	BRPE	RW	0x0	Baud Rate Prescaler Extension 0x00-0x0F: Extend the BRP bit in the **CANBIT** register to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by BRPE (MSBs) and BRP (LSBs).

Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020

Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080

A message transfer is started as soon as there is a write of the message object number to the MNUM field when the TXRQST bit in the **CANIF1MCTL** register is set. With this write operation, the BUSY bit is automatically set to indicate that a transfer between the CAN Interface Registers and the internal message RAM is in progress. After a wait time of 3 to 6 CAN_CLK periods, the transfer between the interface register and the message RAM completes, which then clears the BUSY bit.

CAN IFn Command Request (CANIFnCRQ)

CAN0 base: 0x4004.0000
Offset 0x020
Type RW, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BUSY	reserved									MNUM					
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	BUSY	RO	0	Busy Flag Value Description 0 This bit is cleared when read/write action has finished. 1 This bit is set when a write occurs to the message number in this register.
14:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	MNUM	RW	0x01	Message Number Selects one of the 32 message objects in the message RAM for data transfer. The message objects are numbered from 1 to 32. Value Description 0x00 Reserved 0 is not a valid message number; it is interpreted as 0x20, or object 32. 0x01-0x20 Message Number Indicates specified message object 1 to 32. 0x21-0x3F Reserved Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F.

Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024

Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084

Reading the Command Mask registers provides status for various functions. Writing to the Command Mask registers specifies the transfer direction and selects which buffer registers are the source or target of the data transfer.

Note that when a read from the message object buffer occurs when the WRNRD bit is clear and the CLRINTPND and/or NEWDAT bits are set, the interrupt pending and/or new data flags in the message object buffer are cleared.

CAN IFn Command Mask (CANIFnCMSK)

CAN0 base: 0x4004.0000
Offset 0x024
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								WRNRD	MASK	ARB	CONTROL	CLRINTPND	NEWDAT / TXRQST	DATAA	DATAB
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	WRNRD	RW	0	Write, Not Read Value Description 0 Transfer the data in the CAN message object specified by the MNUM field in the **CANIFnCRQ** register into the **CANIFn** registers. 1 Transfer the data in the **CANIFn** registers to the CAN message object specified by the MNUM field in the **CAN Command Request (CANIFnCRQ)**. **Note:** Interrupt pending and new data conditions in the message buffer can be cleared by reading from the buffer (WRNRD = 0) when the CLRINTPND and/or NEWDAT bits are set.
6	MASK	RW	0	Access Mask Bits Value Description 0 Mask bits unchanged. 1 Transfer IDMASK + DIR + MXTD of the message object into the Interface registers.

Bit/Field	Name	Type	Reset	Description
5	ARB	RW	0	Access Arbitration Bits Value — Description 0 — Arbitration bits unchanged. 1 — Transfer `ID` + `DIR` + `XTD` + `MSGVAL` of the message object into the Interface registers.
4	CONTROL	RW	0	Access Control Bits Value — Description 0 — Control bits unchanged. 1 — Transfer control bits from the **CANIFnMCTL** register into the Interface registers.
3	CLRINTPND	RW	0	Clear Interrupt Pending Bit The function of this bit depends on the configuration of the `WRNRD` bit. Value — Description 0 — If `WRNRD` is clear, the interrupt pending status is transferred from the message buffer into the **CANIFnMCTL** register. If `WRNRD` is set, the `INTPND` bit in the message object remains unchanged. 1 — If `WRNRD` is clear, the interrupt pending status is cleared in the message buffer. Note the value of this bit that is transferred to the **CANIFnMCTL** register always reflects the status of the bits before clearing. If `WRNRD` is set, the `INTPND` bit is cleared in the message object.
2	NEWDAT / TXRQST	RW	0	NEWDAT / TXRQST Bit The function of this bit depends on the configuration of the `WRNRD` bit. Value — Description 0 — If `WRNRD` is clear, the value of the new data status is transferred from the message buffer into the **CANIFnMCTL** register. If `WRNRD` is set, a transmission is not requested. 1 — If `WRNRD` is clear, the new data status is cleared in the message buffer. Note the value of this bit that is transferred to the **CANIFnMCTL** register always reflects the status of the bits before clearing. If `WRNRD` is set, a transmission is requested. Note that when this bit is set, the `TXRQST` bit in the **CANIFnMCTL** register is ignored.

Bit/Field	Name	Type	Reset	Description
1	DATAA	RW	0	Access Data Byte 0 to 3 The function of this bit depends on the configuration of the `WRNRD` bit. Value — Description 0 — Data bytes 0-3 are unchanged. 1 — If `WRNRD` is clear, transfer data bytes 0-3 in **CANIFnDA1** and **CANIFnDA2** to the message object. If `WRNRD` is set, transfer data bytes 0-3 in message object to **CANIFnDA1** and **CANIFnDA2**.
0	DATAB	RW	0	Access Data Byte 4 to 7 The function of this bit depends on the configuration of the `WRNRD` bit as follows: Value — Description 0 — Data bytes 4-7 are unchanged. 1 — If `WRNRD` is clear, transfer data bytes 4-7 in **CANIFnDA1** and **CANIFnDA2** to the message object. If `WRNRD` is set, transfer data bytes 4-7 in message object to **CANIFnDA1** and **CANIFnDA2**.

Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028

Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088

The mask information provided in this register accompanies the data (**CANIFnDAn**), arbitration information (**CANIFnARBn**), and control information (**CANIFnMCTL**) to the message object in the message RAM. The mask is used with the `ID` bit in the **CANIFnARBn** register for acceptance filtering. Additional mask information is contained in the **CANIFnMSK2** register.

CAN IFn Mask 1 (CANIFnMSK1)

CAN0 base: 0x4004.0000
Offset 0x028
Type RW, 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
	MSK															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MSK	RW	0xFFFF	Identifier Mask When using a 29-bit identifier, these bits are used for bits [15:0] of the ID. The `MSK` field in the **CANIFnMSK2** register are used for bits [28:16] of the ID. When using an 11-bit identifier, these bits are ignored.

Value	Description
0	The corresponding identifier field (`ID`) in the message object cannot inhibit the match in acceptance filtering.
1	The corresponding identifier field (`ID`) is used for acceptance filtering.

Register 14: CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C

Register 15: CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C

This register holds extended mask information that accompanies the **CANIFnMSK1** register.

CAN IFn Mask 2 (CANIFnMSK2)

CAN0 base: 0x4004.0000
Offset 0x02C
Type RW, 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	MSK												
Type	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
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	RW	1	Mask Extended Identifier

Value	Description
0	The extended identifier bit (`XTD` in the **CANIFnARB2** register) has no effect on the acceptance filtering.
1	The extended identifier bit `XTD` is used for acceptance filtering.

Bit/Field	Name	Type	Reset	Description
14	MDIR	RW	1	Mask Message Direction

Value	Description
0	The message direction bit (`DIR` in the **CANIFnARB2** register) has no effect for acceptance filtering.
1	The message direction bit `DIR` is used for acceptance filtering.

Bit/Field	Name	Type	Reset	Description
13	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
12:0	MSK	RW	0xFF	Identifier Mask When using a 29-bit identifier, these bits are used for bits [28:16] of the ID. The MSK field in the **CANIFnMSK1** register are used for bits [15:0] of the ID. When using an 11-bit identifier, MSK[12:2] are used for bits [10:0] of the ID. 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 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030

Register 17: CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090

These registers hold the identifiers for acceptance filtering.

CAN IFn Arbitration 1 (CANIFnARB1)

CAN0 base: 0x4004.0000
Offset 0x030
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	ID	RW	0x0000	Message Identifier This bit field is used with the `ID` field in the **CANIFnARB2** register to create the message identifier. When using a 29-bit identifier, bits 15:0 of the **CANIFnARB1** register are [15:0] of the ID, while bits 12:0 of the **CANIFnARB2** register are [28:16] of the ID. When using an 11-bit identifier, these bits are not used.

Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034

Register 19: CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094

These registers hold information for acceptance filtering.

CAN IFn Arbitration 2 (CANIFnARB2)

CAN0 base: 0x4004.0000
Offset 0x034
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSGVAL	XTD	DIR	ID												
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	MSGVAL	RW	0	Message Valid 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	RW	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	RW	0	Message Direction Value Description 0 Receive. When the `TXRQST` bit in the **CANIFnMCTL** register is set, a remote frame with the identifier of this message object is received. On reception of a data frame with matching identifier, that message is stored in this message object. 1 Transmit. When the `TXRQST` bit in the **CANIFnMCTL** register is set, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the `TXRQST` bit of this message object is set (if `RMTEN=1`).
12:0	ID	RW	0x000	Message Identifier This bit field is used with the `ID` field in the **CANIFnARB2** register to create the message identifier. When using a 29-bit identifier, `ID[15:0]` of the **CANIFnARB1** register are [15:0] of the ID, while these bits, `ID[12:0]`, are [28:16] of the ID. When using an 11-bit identifier, `ID[12:2]` are used for bits [10:0] of the ID. The `ID` field in the **CANIFnARB1** register is ignored.

Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038

Register 21: CAN IF2 Message Control (CANIF2MCTL), offset 0x098

This register holds the control information associated with the message object to be sent to the Message RAM.

CAN IFn Message Control (CANIFnMCTL)

CAN0 base: 0x4004.0000
Offset 0x038
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB	reserved			DLC			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	NEWDAT	RW	0	New Data 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	RW	0	Message Lost 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. This bit is only valid for message objects when the `DIR` bit in the **CANIFnARB2** register is clear (receive).
13	INTPND	RW	0	Interrupt Pending 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	RW	0	Use Acceptance Mask Value Description 0 Mask is ignored. 1 Use mask (MSK, MXTD, and MDIR bits in the **CANIFnMSKn** registers) for acceptance filtering.
11	TXIE	RW	0	Transmit Interrupt Enable Value Description 0 The INTPND bit in the **CANIFnMCTL** register is unchanged after a successful transmission of a frame. 1 The INTPND bit in the **CANIFnMCTL** register is set after a successful transmission of a frame.
10	RXIE	RW	0	Receive Interrupt Enable Value Description 0 The INTPND bit in the **CANIFnMCTL** register is unchanged after a successful reception of a frame. 1 The INTPND bit in the **CANIFnMCTL** register is set after a successful reception of a frame.
9	RMTEN	RW	0	Remote Enable Value Description 0 At the reception of a remote frame, the TXRQST bit in the **CANIFnMCTL** register is left unchanged. 1 At the reception of a remote frame, the TXRQST bit in the **CANIFnMCTL** register is set.
8	TXRQST	RW	0	Transmit Request Value Description 0 This message object is not waiting for transmission. 1 The transmission of this message object is requested and is not yet done. **Note:** If the WRNRD and TXRQST bits in the **CANIFnCMSK** register are set, this bit is ignored.

<table>
<tr><th>Bit/Field</th><th>Name</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>7</td><td>EOB</td><td>RW</td><td>0</td><td>End of Buffer
<table>
<tr><th>Value</th><th>Description</th></tr>
<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>
</table>
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.</td></tr>
<tr><td>6:4</td><td>reserved</td><td>RO</td><td>0x0</td><td>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</td></tr>
<tr><td>3:0</td><td>DLC</td><td>RW</td><td>0x0</td><td>Data Length Code
<table>
<tr><th>Value</th><th>Description</th></tr>
<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>
</table>
The DLC field in the <b>CANIFnMCTL</b> 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.</td></tr>
</table>

Register 22: CAN IF1 Data A1 (CANIF1DA1), offset 0x03C

Register 23: CAN IF1 Data A2 (CANIF1DA2), offset 0x040

Register 24: CAN IF1 Data B1 (CANIF1DB1), offset 0x044

Register 25: CAN IF1 Data B2 (CANIF1DB2), offset 0x048

Register 26: CAN IF2 Data A1 (CANIF2DA1), offset 0x09C

Register 27: CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0

Register 28: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4

Register 29: CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8

These registers contain the data to be sent or that has been received. In a CAN data frame, data byte 0 is the first byte to be transmitted or received and data byte 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte is transmitted first.

CAN IFn Data nn (CANIFnDnn)

CAN0 base: 0x4004.0000
Offset 0x03C
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	RW	0x0000	Data The **CANIFnDA1** registers contain data bytes 1 and 0; **CANIFnDA2** data bytes 3 and 2; **CANIFnDB1** data bytes 5 and 4; and **CANIFnDB2** data bytes 7 and 6.

Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100

Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104

The **CANTXRQ1** and **CANTXRQ2** registers hold the TXRQST bits of the 32 message objects. By reading out these bits, the CPU can check which message object has a transmission request pending. The TXRQST bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a remote frame, or (3) the message handler state machine after a successful transmission.

The **CANTXRQ1** register contains the TXRQST bits of the first 16 message objects in the message RAM; the **CANTXRQ2** register contains the TXRQST bits of the second 16 message objects.

CAN Transmission Request n (CANTXRQn)

CAN0 base: 0x4004.0000
Offset 0x100
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
	TXRQST															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	TXRQST	RO	0x0000	Transmission Request Bits

Value	Description
0	The corresponding message object is not waiting for transmission.
1	The transmission of the corresponding message object is requested and is not yet done.

Register 32: CAN New Data 1 (CANNWDA1), offset 0x120

Register 33: CAN New Data 2 (CANNWDA2), offset 0x124

The **CANNWDA1** and **CANNWDA2** registers hold the `NEWDAT` bits of the 32 message objects. By reading these bits, the CPU can check which message object has its data portion updated. The `NEWDAT` bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a data frame, or (3) the message handler state machine after a successful transmission.

The **CANNWDA1** register contains the `NEWDAT` bits of the first 16 message objects in the message RAM; the **CANNWDA2** register contains the `NEWDAT` bits of the second 16 message objects.

CAN New Data n (CANNWDAn)

CAN0 base: 0x4004.0000
Offset 0x120
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
	NEWDAT															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	NEWDAT	RO	0x0000	New Data Bits

Value	Description
0	No new data has been written into the data portion of the corresponding message object by the message handler since the last time this flag was cleared by the CPU.
1	The message handler or the CPU has written new data into the data portion of the corresponding message object.

Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140

Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144

The **CANMSG1INT** and **CANMSG2INT** registers hold the `INTPND` bits of the 32 message objects. By reading these bits, the CPU can check which message object has an interrupt pending. The `INTPND` bit of a specific message object can be changed through two sources: (1) the CPU via the **CANIFnMCTL** register, or (2) the message handler state machine after the reception or transmission of a frame.

This field is also encoded in the **CANINT** register.

The **CANMSG1INT** register contains the `INTPND` bits of the first 16 message objects in the message RAM; the **CANMSG2INT** register contains the `INTPND` bits of the second 16 message objects.

CAN Message n Interrupt Pending (CANMSGnINT)

CAN0 base: 0x4004.0000
Offset 0x140
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
	INTPND															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	INTPND	RO	0x0000	Interrupt Pending Bits Value Description 0 The corresponding message object is not the source of an interrupt. 1 The corresponding message object is the source of an interrupt.

Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160

Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164

The **CANMSG1VAL** and **CANMSG2VAL** registers hold the `MSGVAL` bits of the 32 message objects. By reading these bits, the CPU can check which message object is valid. The message valid bit of a specific message object can be changed with the **CANIFnARB2** register.

The **CANMSG1VAL** register contains the `MSGVAL` bits of the first 16 message objects in the message RAM; the **CANMSG2VAL** register contains the `MSGVAL` bits of the second 16 message objects in the message RAM.

CAN Message n Valid (CANMSGnVAL)

CAN0 base: 0x4004.0000
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.

17 Universal Serial Bus (USB) Controller

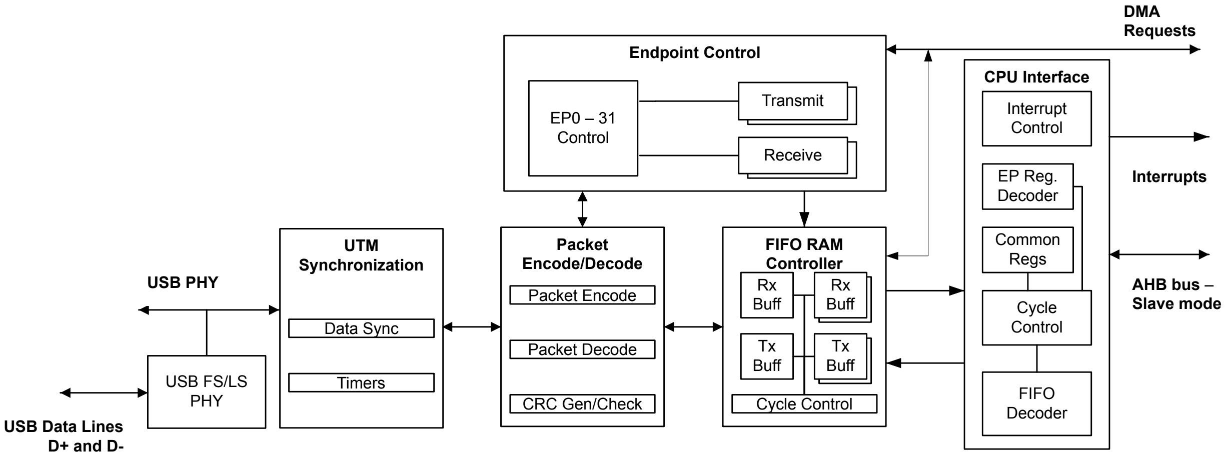
The TM4C1232C3PM USB controller operates as a full-speed function controller during point-to-point communications with USB Host functions. The controller complies with the USB 2.0 standard, which includes SUSPEND and RESUME signaling. 16 endpoints including two hard-wired for control transfers (one endpoint for IN and one endpoint for OUT) plus 14 endpoints defined by firmware along with a dynamic sizable FIFO support multiple packet queueing. µDMA access to the FIFO allows minimal interference from system software. Software-controlled connect and disconnect allows flexibility during USB device startup.

The TM4C1232C3PM USB module has the following features:

- Complies with USB-IF (Implementer's Forum) certification standards
- USB 2.0 full-speed (12 Mbps) operation with integrated PHY
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 16 endpoints
  - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
  - 7 configurable IN endpoints and 7 configurable OUT endpoints
- 4 KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - 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

17.1 Block Diagram

Figure 17-1. USB Module Block Diagram



17.2 Signal Description

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

Table 17-1. USB Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
USB0DM	43	PD4	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
USB0DP	44	PD5	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.

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

17.3 Functional Description

The TM4C1232C3PM USB controller provides the ability for the controller to serve as a Device-only controller. The controller can only be used in Device mode to connect USB-enabled peripherals to the USB controller. For Device mode, the USB controller requires a B connector in the system to provide Device connectivity.

Note: When the USB module is in operation, MOSC must be the clock source, either with or without using the PLL, and the system clock must be at least 20 MHz.

17.3.1 Operation

This section describes the TM4C1232C3PM USB controller's actions. IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and recognition of Start of Frame (SOF) are all described.

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.

17.3.1.1 Endpoints

The USB controller provides two dedicated control endpoints (IN and OUT) and 14 configurable endpoints (7 IN and 7 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 14 endpoints can be configured as control, bulk, interrupt, or isochronous endpoints. They should be treated as 7 configurable IN and 7 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.

17.3.1.2 IN Transactions

Data for IN transactions is handled through the FIFOs attached to the transmit endpoints. The sizes of the FIFOs for the 7 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.

17.3.1.3 OUT Transactions

OUT transactions are handled through the USB controller receive FIFOs. The sizes of the receive FIFOs for the 7 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 (USBRXCSRHn)** 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 **USBRXCSRLn** 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 **USBRXCSRLn** 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 **USBRXCSRHn** 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 (USBRXDPKTBUFDIS)** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

17.3.1.4 Scheduling

The Device has no control over the scheduling of transactions as scheduling is determined by the Host controller. The TM4C1232C3PM 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.

17.3.1.5 Additional Actions

The USB controller responds automatically to certain conditions on the USB bus or actions by the Host controller such as when the USB controller automatically stalls a control transfer or 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 **USBRXMAXPn** 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 DATAEND 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.

17.3.1.6 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.

Important: When configured as a self-powered Device, the USB module meets the response timing and power draw requirements for USB compliance of SUSPEND mode. When configured as a bus-powered Device, the USB can operate in SUSPEND mode but produces a higher power draw than required to be compliant.

17.3.1.7 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.

17.3.1.8 USB RESET

When 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.

17.3.1.9 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.

17.3.2 DMA Operation

The USB peripheral provides an interface connected to the μDMA controller with separate channels for 3 transmit endpoints and 3 receive endpoints. Software selects which endpoints to service with the μDMA channels using the **USB DMA Select (USBDMASEL)** register. The μDMA operation of the USB is enabled through the **USBTXCSRHn** and **USBRXCSRHn** registers, for the TX and RX channels respectively. When μDMA operation is enabled, the USB asserts a μDMA 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 μDMA channel must be configured to operate in Basic mode, and the size of the μDMA transfer must be restricted to whole multiples of the size of the USB FIFO. Both read and write transfers of the USB FIFOs using μDMA must be configured in this manner. For example, if the USB endpoint is configured with a FIFO size of 64 bytes, the μDMA 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 μDMA continues transferring data. If the DMAMOD bit is set, an interrupt is generated only when the entire μDMA transfer is complete. The interrupt occurs on the USB interrupt vector. Therefore, if interrupts are used for USB operation and the μDMA is enabled, the USB interrupt handler must be designed to handle the μDMA completion interrupt.

Care must be taken when using the μDMA 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 17-2. Remainder (MAXLOAD/4)

Value	Description
0	MAXLOAD = 64 bytes

Table 17-2. Remainder (MAXLOAD/4) *(continued)*

Value	Description
1	MAXLOAD = 61 bytes
2	MAXLOAD = 62 bytes
3	MAXLOAD = 63 bytes

Table 17-3. Actual Bytes Read

Value	Description
0	MAXLOAD
1	MAXLOAD+3
2	MAXLOAD+2
3	MAXLOAD+1

Table 17-4. 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 (μDMA)” on page 517 for more details about programming the μDMA controller.

17.4 Initialization and Configuration

To use the USB Controller, the peripheral clock must be enabled via the **RCGCUSB** register (see page 328).

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.

The USB controller provides a method to set the current operating mode of the USB controller. This register should be written with the desired default mode so that the controller can respond to external USB events.

17.4.1 Endpoint Configuration

To start communication, the endpoint registers must first be configured. An endpoint must be configured before enumerating to the Host controller.

The endpoint 0 configuration is limited because it is a fixed-function, fixed-FIFO-size endpoint. 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. 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. 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 2 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.

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.

17.5 Register Map

Table 17-5 on page 1035 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 328). There must be a delay of 3 system clocks after the USB module clock is enabled before any USB module registers are accessed.

Table 17-5. Universal Serial Bus (USB) Controller Register Map

Offset	Name	Type	Reset	Description	See page
0x000	USBFADDR	RW	0x00	USB Device Functional Address	1039
0x001	USBPOWER	RW	0x20	USB Power	1040
0x002	USBTXIS	RO	0x0000	USB Transmit Interrupt Status	1042
0x004	USBRXIS	RO	0x0000	USB Receive Interrupt Status	1044
0x006	USBTXIE	RW	0xFFFF	USB Transmit Interrupt Enable	1045
0x008	USBRXIE	RW	0xFFFE	USB Receive Interrupt Enable	1047
0x00A	USBIS	RO	0x00	USB General Interrupt Status	1048
0x00B	USBIE	RW	0x06	USB Interrupt Enable	1049
0x00C	USBFRAME	RO	0x0000	USB Frame Value	1050
0x00E	USBEPIDX	RW	0x00	USB Endpoint Index	1051
0x00F	USBTEST	RW	0x00	USB Test Mode	1052
0x020	USBFIFO0	RW	0x0000.0000	USB FIFO Endpoint 0	1053
0x024	USBFIFO1	RW	0x0000.0000	USB FIFO Endpoint 1	1053
0x028	USBFIFO2	RW	0x0000.0000	USB FIFO Endpoint 2	1053
0x02C	USBFIFO3	RW	0x0000.0000	USB FIFO Endpoint 3	1053
0x030	USBFIFO4	RW	0x0000.0000	USB FIFO Endpoint 4	1053
0x034	USBFIFO5	RW	0x0000.0000	USB FIFO Endpoint 5	1053
0x038	USBFIFO6	RW	0x0000.0000	USB FIFO Endpoint 6	1053
0x03C	USBFIFO7	RW	0x0000.0000	USB FIFO Endpoint 7	1053

Table 17-5. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x062	USBTXFIFOSZ	RW	0x00	USB Transmit Dynamic FIFO Sizing	1054
0x063	USBRXFIFOSZ	RW	0x00	USB Receive Dynamic FIFO Sizing	1054
0x064	USBTXFIFOADD	RW	0x0000	USB Transmit FIFO Start Address	1055
0x066	USBRXFIFOADD	RW	0x0000	USB Receive FIFO Start Address	1055
0x07A	USBCONTIM	RW	0x5C	USB Connect Timing	1056
0x07D	USBFSEOF	RW	0x77	USB Full-Speed Last Transaction to End of Frame Timing	1057
0x07E	USBLSEOF	RW	0x72	USB Low-Speed Last Transaction to End of Frame Timing	1058
0x102	USBCSRL0	W1C	0x00	USB Control and Status Endpoint 0 Low	1060
0x103	USBCSRH0	W1C	0x00	USB Control and Status Endpoint 0 High	1062
0x108	USBCOUNT0	RO	0x00	USB Receive Byte Count Endpoint 0	1063
0x110	USBTXMAXP1	RW	0x0000	USB Maximum Transmit Data Endpoint 1	1059
0x112	USBTXCSRL1	RW	0x00	USB Transmit Control and Status Endpoint 1 Low	1064
0x113	USBTXCSRH1	RW	0x00	USB Transmit Control and Status Endpoint 1 High	1066
0x114	USBRXMAXP1	RW	0x0000	USB Maximum Receive Data Endpoint 1	1068
0x116	USBRXCSRL1	RW	0x00	USB Receive Control and Status Endpoint 1 Low	1069
0x117	USBRXCSRH1	RW	0x00	USB Receive Control and Status Endpoint 1 High	1072
0x118	USBRXCOUNT1	RO	0x0000	USB Receive Byte Count Endpoint 1	1074
0x120	USBTXMAXP2	RW	0x0000	USB Maximum Transmit Data Endpoint 2	1059
0x122	USBTXCSRL2	RW	0x00	USB Transmit Control and Status Endpoint 2 Low	1064
0x123	USBTXCSRH2	RW	0x00	USB Transmit Control and Status Endpoint 2 High	1066
0x124	USBRXMAXP2	RW	0x0000	USB Maximum Receive Data Endpoint 2	1068
0x126	USBRXCSRL2	RW	0x00	USB Receive Control and Status Endpoint 2 Low	1069
0x127	USBRXCSRH2	RW	0x00	USB Receive Control and Status Endpoint 2 High	1072
0x128	USBRXCOUNT2	RO	0x0000	USB Receive Byte Count Endpoint 2	1074
0x130	USBTXMAXP3	RW	0x0000	USB Maximum Transmit Data Endpoint 3	1059
0x132	USBTXCSRL3	RW	0x00	USB Transmit Control and Status Endpoint 3 Low	1064
0x133	USBTXCSRH3	RW	0x00	USB Transmit Control and Status Endpoint 3 High	1066
0x134	USBRXMAXP3	RW	0x0000	USB Maximum Receive Data Endpoint 3	1068
0x136	USBRXCSRL3	RW	0x00	USB Receive Control and Status Endpoint 3 Low	1069
0x137	USBRXCSRH3	RW	0x00	USB Receive Control and Status Endpoint 3 High	1072
0x138	USBRXCOUNT3	RO	0x0000	USB Receive Byte Count Endpoint 3	1074

Table 17-5. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x140	USBTXMAXP4	RW	0x0000	USB Maximum Transmit Data Endpoint 4	1059
0x142	USBTXCSRL4	RW	0x00	USB Transmit Control and Status Endpoint 4 Low	1064
0x143	USBTXCSRH4	RW	0x00	USB Transmit Control and Status Endpoint 4 High	1066
0x144	USBRXMAXP4	RW	0x0000	USB Maximum Receive Data Endpoint 4	1068
0x146	USBRXCSRL4	RW	0x00	USB Receive Control and Status Endpoint 4 Low	1069
0x147	USBRXCSRH4	RW	0x00	USB Receive Control and Status Endpoint 4 High	1072
0x148	USBRXCOUNT4	RO	0x0000	USB Receive Byte Count Endpoint 4	1074
0x150	USBTXMAXP5	RW	0x0000	USB Maximum Transmit Data Endpoint 5	1059
0x152	USBTXCSRL5	RW	0x00	USB Transmit Control and Status Endpoint 5 Low	1064
0x153	USBTXCSRH5	RW	0x00	USB Transmit Control and Status Endpoint 5 High	1066
0x154	USBRXMAXP5	RW	0x0000	USB Maximum Receive Data Endpoint 5	1068
0x156	USBRXCSRL5	RW	0x00	USB Receive Control and Status Endpoint 5 Low	1069
0x157	USBRXCSRH5	RW	0x00	USB Receive Control and Status Endpoint 5 High	1072
0x158	USBRXCOUNT5	RO	0x0000	USB Receive Byte Count Endpoint 5	1074
0x160	USBTXMAXP6	RW	0x0000	USB Maximum Transmit Data Endpoint 6	1059
0x162	USBTXCSRL6	RW	0x00	USB Transmit Control and Status Endpoint 6 Low	1064
0x163	USBTXCSRH6	RW	0x00	USB Transmit Control and Status Endpoint 6 High	1066
0x164	USBRXMAXP6	RW	0x0000	USB Maximum Receive Data Endpoint 6	1068
0x166	USBRXCSRL6	RW	0x00	USB Receive Control and Status Endpoint 6 Low	1069
0x167	USBRXCSRH6	RW	0x00	USB Receive Control and Status Endpoint 6 High	1072
0x168	USBRXCOUNT6	RO	0x0000	USB Receive Byte Count Endpoint 6	1074
0x170	USBTXMAXP7	RW	0x0000	USB Maximum Transmit Data Endpoint 7	1059
0x172	USBTXCSRL7	RW	0x00	USB Transmit Control and Status Endpoint 7 Low	1064
0x173	USBTXCSRH7	RW	0x00	USB Transmit Control and Status Endpoint 7 High	1066
0x174	USBRXMAXP7	RW	0x0000	USB Maximum Receive Data Endpoint 7	1068
0x176	USBRXCSRL7	RW	0x00	USB Receive Control and Status Endpoint 7 Low	1069
0x177	USBRXCSRH7	RW	0x00	USB Receive Control and Status Endpoint 7 High	1072
0x178	USBRXCOUNT7	RO	0x0000	USB Receive Byte Count Endpoint 7	1074
0x340	USBRXDPKTBUFDIS	RW	0x0000	USB Receive Double Packet Buffer Disable	1075
0x342	USBTXDPKTBUFDIS	RW	0x0000	USB Transmit Double Packet Buffer Disable	1076
0x410	USBDRRIS	RO	0x0000.0000	USB Device RESUME Raw Interrupt Status	1077
0x414	USBDRIM	RW	0x0000.0000	USB Device RESUME Interrupt Mask	1078

Table 17-5. Universal Serial Bus (USB) Controller Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x418	USBDRISC	W1C	0x0000.0000	USB Device RESUME Interrupt Status and Clear	1079
0x450	USBDMASEL	RW	0x0033.2211	USB DMA Select	1080
0xFC0	USBPP	RO	0x0000.1050	USB Peripheral Properties	1082

17.6 Register Descriptions

The TM4C1232C3PM USB controller has Device only capabilities as specified in the `USB0` bit field in the **DC6** register (see page 409).

Register 1: USB Device Functional Address (USBFADDR), offset 0x000

USBFADDR is an 8-bit register that contains the 7-bit address of the Device part of the transaction.

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 1031 for special considerations when writing this register.

USB Device Functional Address (USBFADDR)

Base 0x4005.0000
Offset 0x000
Type RW, reset 0x00

	7	6	5	4	3	2	1	0
	reserved	FUNCADDR						
Type	RO	RW	RW	RW	RW	RW	RW	RW
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.
6:0	FUNCADDR	RW	0x00	Function Address Function Address of Device as received through SET_ADDRESS.

Register 2: USB Power (USBPOWER), offset 0x001

USBPOWER is an 8-bit register used for controlling SUSPEND and RESUME signaling and some basic operational aspects of the USB controller.

USB Power (USBPOWER)

Base 0x4005.0000
Offset 0x001
Type RW, reset 0x20

	7	6	5	4	3	2	1	0
	ISOUP	SOFTCONN	reserved		RESET	RESUME	SUSPEND	PWRDNPHY
Type	RW	RW	RO	RO	RO	RW	RO	RW
Reset	0	0	1	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	ISOUP	RW	0	Isochronous Update Value Description 0 No effect. 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. **Note:** This bit is only valid for isochronous transfers.
6	SOFTCONN	RW	0	Soft Connect/Disconnect Value Description 0 The USB D+/D- lines are tri-stated. 1 The USB D+/D- lines are enabled.
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 Signaling Value Description 0 RESET signaling is not present on the bus. 1 RESET signaling is present on the bus.
2	RESUME	RW	0	RESUME Signaling Value Description 0 Ends RESUME signaling on the bus. 1 Enables RESUME signaling when the Device is in SUSPEND mode. This bit must be cleared by software 10 ms (a maximum of 15 ms) after being set.

Bit/Field	Name	Type	Reset	Description
1	SUSPEND	RO	0	SUSPEND Mode Value Description 0 This bit is cleared when software reads the interrupt register or sets the RESUME bit above. 1 The USB controller is in SUSPEND mode.
0	PWRDNPHY	RW	0	Power Down PHY Value Description 0 No effect. 1 Powers down the internal USB PHY.

Register 3: USB Transmit Interrupt Status (USBTXIS), offset 0x002

Important: This register is read-sensitive. See the register description for details.

USBTXIS is a 16-bit read-only register that indicates which interrupts are currently active for endpoint 0 and the transmit endpoints 1–7. The meaning of the EPn bits in this register is based on the mode of the device. The EP1 through EP7 bits always indicate that the USB controller is sending data; however, the bits refer to IN endpoints. The EP0 bit is special and indicates that either a control IN or control OUT endpoint has generated an interrupt.

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
	reserved								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
15:8	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	EP7	RO	0	TX Endpoint 7 Interrupt Value Description 0 No interrupt. 1 The Endpoint 7 transmit interrupt is asserted.
6	EP6	RO	0	TX Endpoint 6 Interrupt Same description as EP7.
5	EP5	RO	0	TX Endpoint 5 Interrupt Same description as EP7.
4	EP4	RO	0	TX Endpoint 4 Interrupt Same description as EP7.
3	EP3	RO	0	TX Endpoint 3 Interrupt Same description as EP7.
2	EP2	RO	0	TX Endpoint 2 Interrupt Same description as EP7.
1	EP1	RO	0	TX Endpoint 1 Interrupt Same description as EP7.

Bit/Field	Name	Type	Reset	Description
0	EP0	RO	0	TX and RX Endpoint 0 Interrupt Value Description 0 No interrupt. 1 The Endpoint 0 transmit and receive interrupt is asserted.

Register 4: USB Receive Interrupt Status (USBRXIS), offset 0x004

Important: This register is read-sensitive. See the register description for details.

USBRXIS is a 16-bit read-only register that indicates which of the interrupts for receive endpoints 1–7 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.

USB Receive Interrupt Status (USBRXIS)

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
	reserved								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
15:8	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	EP7	RO	0	RX Endpoint 7 Interrupt Value Description 0 No interrupt. 1 The Endpoint 7 transmit interrupt is asserted.
6	EP6	RO	0	RX Endpoint 6 Interrupt Same description as EP7.
5	EP5	RO	0	RX Endpoint 5 Interrupt Same description as EP7.
4	EP4	RO	0	RX Endpoint 4 Interrupt Same description as EP7.
3	EP3	RO	0	RX Endpoint 3 Interrupt Same description as EP7.
2	EP2	RO	0	RX Endpoint 2 Interrupt Same description as EP7
1	EP1	RO	0	RX Endpoint 1 Interrupt Same description as EP7.
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

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.

USB Transmit Interrupt Enable (USBTXIE)

Base 0x4005.0000
Offset 0x006
Type RW, reset 0xFFFF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
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	EP7	RW	1	TX Endpoint 7 Interrupt Enable Value Description 0 The EP7 transmit interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the EP7 bit in the **USBTXIS** register is set.
6	EP6	RW	1	TX Endpoint 6 Interrupt Enable Same description as EP7.
5	EP5	RW	1	TX Endpoint 5 Interrupt Enable Same description as EP7.
4	EP4	RW	1	TX Endpoint 4 Interrupt Enable Same description as EP7.
3	EP3	RW	1	TX Endpoint 3 Interrupt Enable Same description as EP7.
2	EP2	RW	1	TX Endpoint 2 Interrupt Enable Same description as EP7.
1	EP1	RW	1	TX Endpoint 1 Interrupt Enable Same description as EP7.

Bit/Field	Name	Type	Reset	Description
0	EP0	RW	1	TX and RX Endpoint 0 Interrupt Enable Value Description 0 The EP0 transmit and receive interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the EP0 bit in the **USBTXIS** register is set.

Register 6: USB Receive Interrupt Enable (USBRXIE), offset 0x008

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.

USB Receive Interrupt Enable (USBRXIE)

Base 0x4005.0000
Offset 0x008
Type RW, reset 0xFFFE

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								EP7	EP6	EP5	EP4	EP3	EP2	EP1	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0

Bit/Field	Name	Type	Reset	Description
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	EP7	RW	1	RX Endpoint 7 Interrupt Enable Value Description 0 The EP7 receive interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the EP7 bit in the **USBRXIS** register is set.
6	EP6	RW	1	RX Endpoint 6 Interrupt Enable Same description as EP7.
5	EP5	RW	1	RX Endpoint 5 Interrupt Enable Same description as EP7.
4	EP4	RW	1	RX Endpoint 4 Interrupt Enable Same description as EP7.
3	EP3	RW	1	RX Endpoint 3 Interrupt Enable Same description as EP7.
2	EP2	RW	1	RX Endpoint 2 Interrupt Enable Same description as EP7.
1	EP1	RW	1	RX Endpoint 1 Interrupt Enable Same description as EP7.
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

Important: This register is read-sensitive. See the register description for details.

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.

USB General Interrupt Status (USBIS)

Base 0x4005.0000
Offset 0x00A
Type RO, reset 0x00

	7	6	5	4	3	2	1	0
	reserved				SOF	RESET	RESUME	SUSPEND
Type	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7:4	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 0 No interrupt. 1 A new frame has started.
2	RESET	RO	0	RESET Signaling Detected Value Description 0 No interrupt. 1 RESET signaling has been detected on the bus.
1	RESUME	RO	0	RESUME Signaling Detected Value Description 0 No interrupt. 1 RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode. 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 Signaling Detected Value Description 0 No interrupt. 1 SUSPEND signaling has been detected on the bus.

Register 8: USB Interrupt Enable (USBIE), offset 0x00B

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.

USB Interrupt Enable (USBIE)

Base 0x4005.0000
Offset 0x00B
Type RW, reset 0x06

	7	6	5	4	3	2	1	0
	reserved				SOF	RESET	RESUME	SUSPEND
Type	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	1	1	0

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	RW	0	Enable Start-of-Frame Interrupt Value Description 0 The `SOF` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `SOF` bit in the **USBIS** register is set.
2	RESET	RW	1	Enable RESET Interrupt Value Description 0 The `RESET` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `RESET` bit in the **USBIS** register is set.
1	RESUME	RW	1	Enable RESUME Interrupt Value Description 0 The `RESUME` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `RESUME` bit in the **USBIS** register is set.
0	SUSPEND	RW	0	Enable SUSPEND Interrupt Value Description 0 The `SUSPEND` interrupt is suppressed and not sent to the interrupt controller. 1 An interrupt is sent to the interrupt controller when the `SUSPEND` bit in the **USBIS** register is set.

Register 9: USB Frame Value (USBFRAME), offset 0x00C

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					FRAME										
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
15:11	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

Each endpoint's buffer can be accessed by configuring a FIFO size and starting address. The **USBEPIDX** 8-bit register is used with the **USBTXFIFOSZ**, **USBRXFIFOSZ**, **USBTXFIFOADD**, and **USBRXFIFOADD** registers.

USB Endpoint Index (USBEPIDX)

Base 0x4005.0000
Offset 0x00E
Type RW, reset 0x00

	7	6	5	4	3	2	1	0
	reserved				EPIDX			
Type	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

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	RW	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 0x7 corresponds to Endpoint 7.

Register 11: USB Test Mode (USBTEST), offset 0x00F

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.

Note: Only one of these bits should be set at any time.

USB Test Mode (USBTEST)

Base 0x4005.0000
Offset 0x00F
Type RW, reset 0x00

	7	6	5	4	3	2	1	0
	reserved	FIFOACC	FORCEFS	reserved				
Type	RO	RW1S	RW	RO	RO	RO	RO	RO
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	FIFOACC	RW1S	0	FIFO Access Value Description 0 No effect. 1 Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. This bit is cleared automatically.
5	FORCEFS	RW	0	Force Full-Speed Mode Value Description 0 The USB controller operates at Low Speed. 1 Forces the USB controller into Full-Speed mode upon receiving a USB RESET.
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

Important: This register is read-sensitive. See the register description for details.

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.

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 1030). 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 1–7, the associated FIFO is completely flushed.

USB FIFO Endpoint n (USBFIFOn)

Base 0x4005.0000
Offset 0x020
Type RW, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:0	EPDATA	RW	0x0000.0000	Endpoint Data Writing to this register loads the data into the Transmit FIFO and reading unloads data from the Receive FIFO.

Register 20: USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062

Register 21: USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063

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 Dynamic FIFO Sizing (USBnXFIFOSZ)

Base 0x4005.0000
Offset 0x062
Type RW, reset 0x00

	7	6	5	4	3	2	1	0
	reserved			DPB	SIZE			
Type	RO	RO	RO	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

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	RW	0	Double Packet Buffer Support
3:0	SIZE	RW	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.

DPB values:

Value	Description
0	Only single-packet buffering is supported.
1	Double-packet buffering is supported.

SIZE values:

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 22: USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064

Register 23: USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066

USBTXFIFOADD and **USBRXFIFOADD** are 16-bit registers that control the start address of the selected transmit and receive endpoint FIFOs.

USB Transmit FIFO Start Address (USBnXFIFOADD)

Base 0x4005.0000
Offset 0x064
Type RW, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							ADDR								
Type	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
15:9	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.
8:0	ADDR	RW	0x00	Transmit/Receive Start Address Start address of the endpoint FIFO.

Value	Start Address
0x0	0
0x1	8
0x2	16
0x3	24
0x4	32
0x5	40
0x6	48
0x7	56
0x8	64
...	...
0x1FF	4095

Register 24: USB Connect Timing (USBCONTIM), offset 0x07A

This 8-bit configuration register specifies connection delay.

USB Connect Timing (USBCONTIM)

Base 0x4005.0000
Offset 0x07A
Type RW, reset 0x5C

	7	6	5	4	3	2	1	0
	WTCON				WTID			
Type	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	1	0	1	1	1	0	0

Bit/Field	Name	Type	Reset	Description
7:4	WTCON	RW	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 µs.
3:0	WTID	RW	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 25: USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF), offset 0x07D

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.

USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF)

Base 0x4005.0000
Offset 0x07D
Type RW, reset 0x77

	7	6	5	4	3	2	1	0
	FSEOFG							
Type	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	1	1	1	0	1	1	1

Bit/Field	Name	Type	Reset	Description
7:0	FSEOFG	RW	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 µs.

Register 26: USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF), offset 0x07E

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.

USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF)

Base 0x4005.0000
Offset 0x07E
Type RW, reset 0x72

	7	6	5	4	3	2	1	0
	LSEOFG							
Type	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	1	1	1	0	0	1	0

Bit/Field	Name	Type	Reset	Description
7:0	LSEOFG	RW	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 μs. The default corresponds to 121.6 μs.

Register 27: USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110

Register 28: USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120

Register 29: USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130

Register 30: USB Maximum Transmit Data Endpoint 4 (USBTXMAXP4), offset 0x140

Register 31: USB Maximum Transmit Data Endpoint 5 (USBTXMAXP5), offset 0x150

Register 32: USB Maximum Transmit Data Endpoint 6 (USBTXMAXP6), offset 0x160

Register 33: USB Maximum Transmit Data Endpoint 7 (USBTXMAXP7), offset 0x170

The **USBTXMAXPn** 16-bit register defines the maximum amount of data that can be transferred through the transmit endpoint in a single operation.

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 **USBTXCSRLn**) after writing the new value to this register.

Note: **USBTXMAXPn** must be set to an even number of bytes for proper interrupt generation in µDMA Basic Mode.

USB Maximum Transmit Data Endpoint n (USBTXMAXPn)

Base 0x4005.0000
Offset 0x110
Type RW, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					MAXLOAD										
Type	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
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	RW	0x000	Maximum Payload This field specifies the maximum payload in bytes per transaction.

Register 34: USB Control and Status Endpoint 0 Low (USBCSRL0), offset 0x102

USBCSRL0 is an 8-bit register that provides control and status bits for endpoint 0.

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	RW	RO	RW	RW	RW	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	RW	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	RW	0	Data End Value Description 0 No effect. 1 Set this bit in the following situations: ■ 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	RW	0	Endpoint Stalled Value Description 0 A STALL handshake has not been transmitted. 1 A STALL handshake has been transmitted. Software must clear this bit.
1	TXRDY	RW	0	Transmit Packet Ready Value Description 0 No transmit packet is ready. 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. This bit is cleared automatically when the data packet has been transmitted.
0	RXRDY	RO	0	Receive Packet Ready Value Description 0 No data packet has been received. 1 A data packet has been received. The `EP0` bit in the **USBTXIS** register is also set in this situation. This bit is cleared by writing a 1 to the `RXRDYC` bit.

Register 35: USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103

USBSR0H is an 8-bit register that provides control and status bits for endpoint 0.

USB Control and Status Endpoint 0 High (USBCSRH0)

Base 0x4005.0000
Offset 0x103
Type W1C, reset 0x00

	7	6	5	4	3	2	1	0
	reserved							FLUSH
Type	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0

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	RW	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 is clear and RXRDY is set. At other times, it may cause data to be corrupted.

Register 36: USB Receive Byte Count Endpoint 0 (USBCOUNT0), offset 0x108

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.

USB Receive Byte Count Endpoint 0 (USBCOUNT0)

Base 0x4005.0000
Offset 0x108
Type RO, reset 0x00

	7	6	5	4	3	2	1	0
	reserved				COUNT			
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	reserved	RO	0	Software should not rely on the value of 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 37: USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1), offset 0x112

Register 38: USB Transmit Control and Status Endpoint 2 Low (USBTXCSRL2), offset 0x122

Register 39: USB Transmit Control and Status Endpoint 3 Low (USBTXCSRL3), offset 0x132

Register 40: USB Transmit Control and Status Endpoint 4 Low (USBTXCSRL4), offset 0x142

Register 41: USB Transmit Control and Status Endpoint 5 Low (USBTXCSRL5), offset 0x152

Register 42: USB Transmit Control and Status Endpoint 6 Low (USBTXCSRL6), offset 0x162

Register 43: USB Transmit Control and Status Endpoint 7 Low (USBTXCSRL7), offset 0x172

USBTXCSRLn is an 8-bit register that provides control and status bits for transfers through the currently selected transmit endpoint.

USB Transmit Control and Status Endpoint n Low (USBTXCSRLn)

Base 0x4005.0000
Offset 0x112
Type RW, reset 0x00

	7	6	5	4	3	2	1	0
	reserved	CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
Type	RO	RW	RW	RW	RW	RW	RW	RW
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	RW	0	Clear Data Toggle Writing a 1 to this bit clears the DT bit in the **USBTXCSRHn** register.
5	STALLED	RW	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.

Bit/Field	Name	Type	Reset	Description
4	STALL	RW	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	RW	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 clear. At other times, it may cause data to be corrupted.
2	UNDRN	RW	0	Underrun Value Description 0 No underrun. 1 An IN token has been received when TXRDY is not set. Software must clear this bit.
1	FIFONE	RW	0	FIFO Not Empty Value Description 0 The FIFO is empty. 1 At least one packet is in the transmit FIFO.
0	TXRDY	RW	0	Transmit Packet Ready Value Description 0 No transmit packet is ready. 1 Software sets this bit after loading a data packet into the TX FIFO. 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.

Register 44: USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1), offset 0x113

Register 45: USB Transmit Control and Status Endpoint 2 High (USBTXCSRH2), offset 0x123

Register 46: USB Transmit Control and Status Endpoint 3 High (USBTXCSRH3), offset 0x133

Register 47: USB Transmit Control and Status Endpoint 4 High (USBTXCSRH4), offset 0x143

Register 48: USB Transmit Control and Status Endpoint 5 High (USBTXCSRH5), offset 0x153

Register 49: USB Transmit Control and Status Endpoint 6 High (USBTXCSRH6), offset 0x163

Register 50: USB Transmit Control and Status Endpoint 7 High (USBTXCSRH7), offset 0x173

USBTXCSRHn is an 8-bit register that provides additional control for transfers through the currently selected transmit endpoint.

USB Transmit Control and Status Endpoint n High (USBTXCSRHn)

Base 0x4005.0000
Offset 0x113
Type RW, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	reserved	
Type	RW	RW	RW	RW	RW	RW	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOSET	RW	0	Auto Set Value Description 0 The TXRDY bit must be set manually. 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.
6	ISO	RW	0	Isochronous Transfers Value Description 0 Enables the transmit endpoint for bulk or interrupt transfers. 1 Enables the transmit endpoint for isochronous transfers.

Bit/Field	Name	Type	Reset	Description
5	MODE	RW	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 where the same endpoint FIFO is used for both transmit and receive transactions.
4	DMAEN	RW	0	DMA Request Enable Value Description 0 Disables the DMA request for the transmit endpoint. 1 Enables the DMA request for the transmit endpoint. **Note:** 3 TX and 3 RX endpoints can be connected to the μDMA module. If this bit is set for a particular endpoint, the DMAATX, DMABTX, or DMACTX field in the **USB DMA Select (USBDMASEL)** register must be programmed correspondingly.
3	FDT	RW	0	Force Data Toggle Value Description 0 No effect. 1 Forces the endpoint DT 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.
2	DMAMOD	RW	0	DMA Request Mode Value Description 0 An interrupt is generated after every DMA packet transfer. 1 An interrupt is generated only after the entire DMA transfer is complete. **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
1:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 51: USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114

Register 52: USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124

Register 53: USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134

Register 54: USB Maximum Receive Data Endpoint 4 (USBRXMAXP4), offset 0x144

Register 55: USB Maximum Receive Data Endpoint 5 (USBRXMAXP5), offset 0x154

Register 56: USB Maximum Receive Data Endpoint 6 (USBRXMAXP6), offset 0x164

Register 57: USB Maximum Receive Data Endpoint 7 (USBRXMAXP7), offset 0x174

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.

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 µDMA Basic mode.

USB Maximum Receive Data Endpoint n (USBRXMAXPn)

Base 0x4005.0000
Offset 0x114
Type RW, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					MAXLOAD										
Type	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
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	RW	0x000	Maximum Payload The maximum payload in bytes per transaction.

Register 58: USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1), offset 0x116

Register 59: USB Receive Control and Status Endpoint 2 Low (USBRXCSRL2), offset 0x126

Register 60: USB Receive Control and Status Endpoint 3 Low (USBRXCSRL3), offset 0x136

Register 61: USB Receive Control and Status Endpoint 4 Low (USBRXCSRL4), offset 0x146

Register 62: USB Receive Control and Status Endpoint 5 Low (USBRXCSRL5), offset 0x156

Register 63: USB Receive Control and Status Endpoint 6 Low (USBRXCSRL6), offset 0x166

Register 64: USB Receive Control and Status Endpoint 7 Low (USBRXCSRL7), offset 0x176

USBRXCSRLn is an 8-bit register that provides control and status bits for transfers through the currently selected receive endpoint.

USB Receive Control and Status Endpoint n Low (USBRXCSRLn)

Base 0x4005.0000
Offset 0x116
Type RW, reset 0x00

	7	6	5	4	3	2	1	0
	CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
Type	W1C	RW	RW	RW	RO	RW	RO	RW
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	CLRDT	W1C	0	Clear Data Toggle Writing a 1 to this bit clears the `DT` bit in the **USBRXCSRHn** register.
6	STALLED	RW	0	Endpoint Stalled Value Description 0 A STALL handshake has not been transmitted. 1 A STALL handshake has been transmitted. Software must clear this bit.
5	STALL	RW	0	Send STALL Value Description 0 No effect. 1 Issues a STALL handshake. Software must clear this bit to terminate the STALL condition. **Note:** This bit has no effect where the endpoint is being used for isochronous transfers.

Bit/Field	Name	Type	Reset	Description
4	FLUSH	RW	0	Flush FIFO Value Description 0 No effect. 1 Flushes the next packet from the endpoint receive FIFO. The FIFO pointer is reset and the `RXRDY` bit is cleared. 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 `RXRDY` bit is cleared. 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 `RXRDY` bit is set. At other times, it may cause data to be corrupted.
3	DATAERR	RO	0	Data Error Value Description 0 Normal operation. 1 Indicates that `RXRDY` is set and the data packet has a CRC or bit-stuff error. This bit is cleared when `RXRDY` is cleared. **Note:** This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.
2	OVER	RW	0	Overrun Value Description 0 No overrun error. 1 Indicates that an OUT packet cannot be loaded into the receive FIFO. Software must clear this bit. **Note:** This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.
1	FULL	RO	0	FIFO Full Value Description 0 The receive FIFO is not full. 1 No more packets can be loaded into the receive FIFO.

Bit/Field	Name	Type	Reset	Description
0	RXRDY	RW	0	Receive Packet Ready Value Description 0 No data packet has been received. 1 A data packet has been received. The EPn bit in the **USBRXIS** register is also set in this situation. If the AUTOCLR bit in the **USBRXCSRHn** register is set, then the this bit is automatically cleared when a packet of **USBRXMAXPn** 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.

Register 65: USB Receive Control and Status Endpoint 1 High (USBRXCSRH1), offset 0x117

Register 66: USB Receive Control and Status Endpoint 2 High (USBRXCSRH2), offset 0x127

Register 67: USB Receive Control and Status Endpoint 3 High (USBRXCSRH3), offset 0x137

Register 68: USB Receive Control and Status Endpoint 4 High (USBRXCSRH4), offset 0x147

Register 69: USB Receive Control and Status Endpoint 5 High (USBRXCSRH5), offset 0x157

Register 70: USB Receive Control and Status Endpoint 6 High (USBRXCSRH6), offset 0x167

Register 71: USB Receive Control and Status Endpoint 7 High (USBRXCSRH7), offset 0x177

USBRXCSRHn is an 8-bit register that provides additional control and status bits for transfers through the currently selected receive endpoint.

USB Receive Control and Status Endpoint n High (USBRXCSRHn)

Base 0x4005.0000
Offset 0x117
Type RW, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD	reserved		
Type	RW	RW	RW	RW	RW	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOCL	RW	0	Auto Clear

Value	Description
0	No effect.
1	Enables the RXRDY bit to be automatically cleared when a packet of **USBRXMAXPn** bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY 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 MAXLOAD field in the **USBRXMAXPn** register, see “DMA Operation” on page 1033.

Bit/Field	Name	Type	Reset	Description
6	ISO	RW	0	Isochronous Transfers Value Description 0 Enables the receive endpoint for isochronous transfers. 1 Enables the receive endpoint for bulk/interrupt transfers.
5	DMAEN	RW	0	DMA Request Enable Value Description 0 Disables the µDMA request for the receive endpoint. 1 Enables the µDMA request for the receive endpoint. **Note:** 3 TX and 3 RX endpoints can be connected to the µDMA module. If this bit is set for a particular endpoint, the DMAARX, DMABRX, or DMACRX field in the **USB DMA Select (USBDMASEL)** register must be programmed correspondingly.
4	DISNYET / PIDERR	RW	0	Disable NYET / PID Error Value Description 0 No effect. 1 *For bulk or interrupt transactions:* 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. *For isochronous transactions:* Indicates a PID error in the received packet.
3	DMAMOD	RW	0	DMA Request Mode Value Description 0 An interrupt is generated after every µDMA packet transfer. 1 An interrupt is generated only after the entire µDMA transfer is complete. **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
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 72: USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118

Register 73: USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128

Register 74: USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138

Register 75: USB Receive Byte Count Endpoint 4 (USBRXCOUNT4), offset 0x148

Register 76: USB Receive Byte Count Endpoint 5 (USBRXCOUNT5), offset 0x158

Register 77: USB Receive Byte Count Endpoint 6 (USBRXCOUNT6), offset 0x168

Register 78: USB Receive Byte Count Endpoint 7 (USBRXCOUNT7), offset 0x178

Note: The value returned changes as the FIFO is unloaded and is only valid while the RXRDY bit in the **USBRXCSRLn** register is set.

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 n (USBRXCOUNTn)

Base 0x4005.0000
Offset 0x118
Type RO, reset 0x0000

	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	0	0	0

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 79: USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS), offset 0x340

USBRXDPKTBUFDIS 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 1030).

USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS)

Base 0x4005.0000
Offset 0x340
Type RW, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								EP7	EP6	EP5	EP4	EP3	EP2	EP1	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
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	EP7	RW	0	EP7 RX Double-Packet Buffer Disable Value Description 0 Disables double-packet buffering. 1 Enables double-packet buffering.
6	EP6	RW	0	EP6 RX Double-Packet Buffer Disable Same description as EP7.
5	EP5	RW	0	EP5 RX Double-Packet Buffer Disable Same description as EP7.
4	EP4	RW	0	EP4 RX Double-Packet Buffer Disable Same description as EP7.
3	EP3	RW	0	EP3 RX Double-Packet Buffer Disable Same description as EP7.
2	EP2	RW	0	EP2 RX Double-Packet Buffer Disable Same description as EP7.
1	EP1	RW	0	EP1 RX Double-Packet Buffer Disable Same description as EP7.
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 80: USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS), offset 0x342

USBTXDPKTBUFDIS 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 1029).

USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS)

Base 0x4005.0000
Offset 0x342
Type RW, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								EP7	EP6	EP5	EP4	EP3	EP2	EP1	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
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	EP7	RW	0	EP7 TX Double-Packet Buffer Disable Value Description 0 Disables double-packet buffering. 1 Enables double-packet buffering.
6	EP6	RW	0	EP6 TX Double-Packet Buffer Disable Same description as EP7.
5	EP5	RW	0	EP5 TX Double-Packet Buffer Disable Same description as EP7.
4	EP4	RW	0	EP4 TX Double-Packet Buffer Disable Same description as EP7.
3	EP3	RW	0	EP3 TX Double-Packet Buffer Disable Same description as EP7.
2	EP2	RW	0	EP2 TX Double-Packet Buffer Disable Same description as EP7.
1	EP1	RW	0	EP1 TX Double-Packet Buffer Disable Same description as EP7.
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 81: USB Device RESUME Raw Interrupt Status (USBDRRIS), offset 0x410

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.

USB Device RESUME Raw Interrupt Status (USBDRRIS)

Base 0x4005.0000
Offset 0x410
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															RESUME
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	RESUME	RO	0	RESUME Interrupt Status Value Description 0 An interrupt has not occurred. 1 A RESUME status has been detected. This bit is cleared by writing a 1 to the `RESUME` bit in the **USBDRISC** register.

Register 82: USB Device RESUME Interrupt Mask (USBDRIM), offset 0x414

The **USBDRIM** 32-bit register is the masked interrupt status register. 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.

USB Device RESUME Interrupt Mask (USBDRIM)

Base 0x4005.0000
Offset 0x414
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															RESUME
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of 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	RW	0	RESUME Interrupt Mask

Value	Description
0	A detected RESUME does not affect the interrupt status.
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 `SUSPEND` bit in the **USBIS** register is set).

Register 83: USB Device RESUME Interrupt Status and Clear (USBDRISC), offset 0x418

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.

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	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	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	RW1C	0	RESUME Interrupt Status and Clear

Value	Description
0	No interrupt has occurred or the interrupt is masked.
1	The RESUME bits in the **USBDRRIS** and **USBDRCIM** registers are set, providing an interrupt to the interrupt controller.

This bit is cleared by writing a 1. Clearing this bit also clears the RESUME bit in the **USBDRCRIS** register.

Register 84: USB DMA Select (USBDMASEL), offset 0x450

This 32-bit register specifies which endpoints are mapped to the 6 allocated μDMA channels, see Table 8-1 on page 519 for more information on channel assignments.

USB DMA Select (USBDMASEL)

Base 0x4005.0000
Offset 0x450
Type RW, 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	RW	RW	RW	RW	RW	RW	RW	RW
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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
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	RW	0x3	DMA C TX Select Specifies the TX mapping of the third USB endpoint on μDMA channel 5 (primary assignment).

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 - 0xF	reserved

Bit/Field	Name	Type	Reset	Description
19:16	DMACRX	RW	0x3	DMA C RX Select Specifies the RX and TX mapping of the third USB endpoint on µDMA channel 4 (primary assignment). 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 - 0xF reserved
15:12	DMABTX	RW	0x2	DMA B TX Select Specifies the TX mapping of the second USB endpoint on µDMA channel 3 (primary assignment). Same bit definitions as the DMACTX field.
11:8	DMABRX	RW	0x2	DMA B RX Select Specifies the RX mapping of the second USB endpoint on µDMA channel 2 (primary assignment). Same bit definitions as the DMACRX field.
7:4	DMAATX	RW	0x1	DMA A TX Select Specifies the TX mapping of the first USB endpoint on µDMA channel 1 (primary assignment). Same bit definitions as the DMACTX field.
3:0	DMAARX	RW	0x1	DMA A RX Select Specifies the RX mapping of the first USB endpoint on µDMA channel 0 (primary assignment). Same bit definitions as the DMACRX field.

Register 85: USB Peripheral Properties (USBPP), offset 0xFC0

The **USBPP** register provides information regarding the properties of the USB module.

USB Peripheral Properties (USBPP)

Base 0x4005.0000
Offset 0xFC0
Type RO, reset 0x0000.1050

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ECNT								USB		reserved	PHY	TYPE			
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	0	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	ECNT	RO	0x10	Endpoint Count This field indicates the hex value for the number of endpoints provided.
7:6	USB	RO	0x1	USB Capability Value Description 0x0 NA USB is not present. 0x1 DEVICE Device Only 0x2 HOST Device or Host 0x3 OTG Device, Host, or OTG
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	PHY	RO	0x1	PHY Present Value Description 0 A PHY is not integrated with the USB MAC. 1 A PHY is integrated with the USB MAC.
3:0	TYPE	RO	0x0	Controller Type Value Description 0x0 The first-generation USB controller. 0x1 - 0xF Reserved

18 Analog Comparators

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

Note: Not all comparators have the option to drive an output pin. See "Signal Description" on page 1084 for more information.

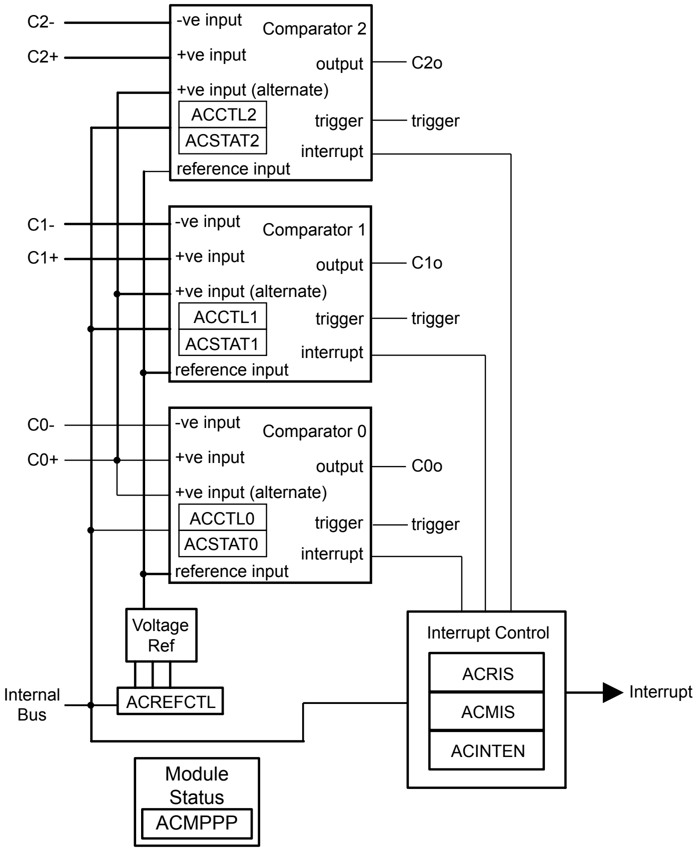
The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board. In addition, the comparator can signal the application via interrupts or trigger the start of a sample sequence in the ADC. The interrupt generation and ADC triggering logic is separate and independent. This flexibility means, for example, that an interrupt can be generated on a rising edge and the ADC triggered on a falling edge.

The TM4C1232C3PM microcontroller provides two 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

18.1 Block Diagram

Figure 18-1. Analog Comparator Module Block Diagram



Note: This block diagram depicts the maximum number of analog comparators and comparator outputs for the family of microcontrollers; the number for this specific device may vary. See page 1097 for what is included on this device.

18.2 Signal Description

The following table lists the external signals of the Analog Comparators and describes the function of each. The Analog Comparator output signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the Analog Comparator signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 603) should be set to choose the Analog Comparator function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 620) 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 581.

Table 18-1. Analog Comparators Signals (64LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
C0+	14	PC6	I	Analog	Analog comparator 0 positive input.

Table 18-1. Analog Comparators Signals (64LQFP) ***(continued)***

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`C0-`	13	PC7	I	Analog	Analog comparator 0 negative input.
`C0o`	28	PF0 (9)	O	TTL	Analog comparator 0 output.
`C1+`	15	PC5	I	Analog	Analog comparator 1 positive input.
`C1-`	16	PC4	I	Analog	Analog comparator 1 negative input.
`C1o`	29	PF1 (9)	O	TTL	Analog comparator 1 output.

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

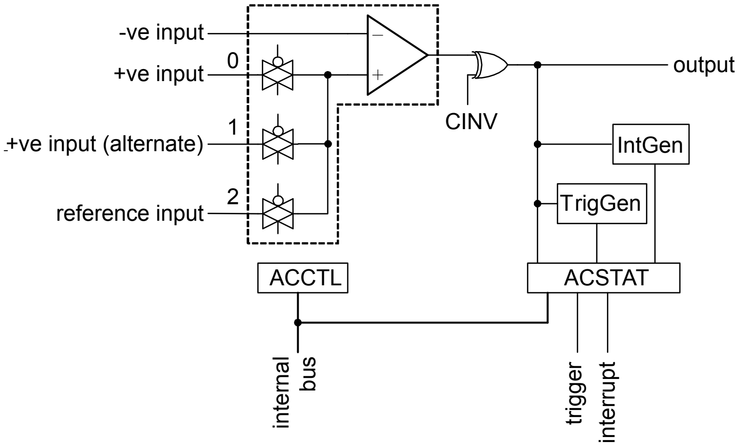
18.3 Functional Description

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 18-2 on page 1085, the input source for VIN- is an external input, `Cn-`, where n is the analog comparator number. In addition to an external input, `Cn+`, input sources for VIN+ can be the C0+ or an internal reference, V_{IREF}.

Figure 18-2. Structure of Comparator Unit



A comparator is configured through two status/control registers, **Analog Comparator Control (ACCTL)** and **Analog Comparator Status (ACSTAT)**. The internal reference is configured through one control register, **Analog Comparator Reference Voltage Control (ACREFCTL)**. Interrupt status and control are configured through three registers, **Analog Comparator Masked Interrupt Status (ACMIS)**, **Analog Comparator Raw Interrupt Status (ACRIS)**, and **Analog Comparator Interrupt Enable (ACINTEN)**.

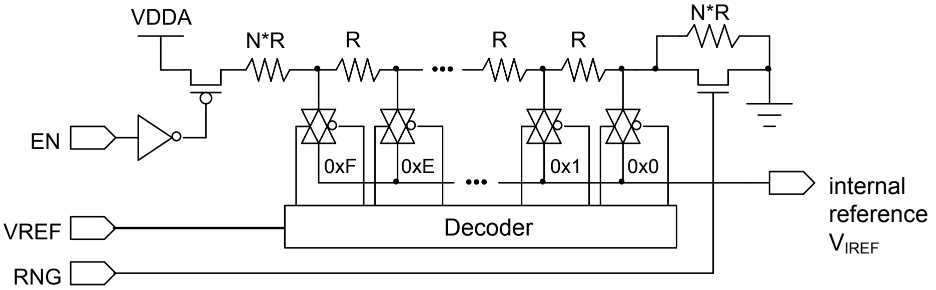
Typically, the comparator output is used internally to generate an interrupt as controlled by the `ISEN` bit in the **ACCTL** register. The output may also be used to drive one of the external pins (`Cno`), or generate an analog-to-digital converter (ADC) trigger.

Important: The ASRCP bits in the **ACCTL** register must be set before using the analog comparators.

18.3.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 18-3 on page 1086. The internal reference is controlled by a single configuration register (**ACREFCTL**).

Figure 18-3. Comparator Internal Reference Structure



Note: In the figure above, N*R represents a multiple of the R value that produces the results specified in Table 18-2 on page 1086.

The internal reference can be programmed in one of two modes (low range or high range) depending on the RNG bit in the **ACREFCTL** register. When RNG is clear, the internal reference is in high-range mode, and when RNG is set the internal reference is in low-range mode.

In each range, the internal reference, V_{IREF}, has 16 preprogrammed thresholds or step values. The threshold to be used to compare the external input voltage against is selected using the VREF field in the **ACREFCTL** register.

In the high-range mode, the V_{IREF} threshold voltages start at the ideal high-range starting voltage of V_{DDA}/4.2 and increase in ideal constant voltage steps of V_{DDA}/29.4.

In the low-range mode, the V_{IREF} threshold voltages start at 0 V and increase in ideal constant voltage steps of V_{DDA}/22.12. The ideal V_{IREF} step voltages for each mode and their dependence on the RNG and VREF fields are summarized in Table 18-2.

Table 18-2. Internal Reference Voltage and ACREFCTL Field Values

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	**RNG Bit Value**	
EN=0	RNG=X	0 V (GND) for any value of VREF. It is recommended that RNG=1 and VREF=0 to minimize noise on the reference ground.
EN=1	RNG=0	V_{IREF} High Range: 16 voltage threshold values indexed by VREF = 0x0 .. 0xF Ideal starting voltage (VREF=0): V_{DDA} / 4.2 Ideal step size: V_{DDA}/ 29.4 Ideal V_{IREF} threshold values: V_{IREF} (VREF) = V_{DDA} / 4.2 + VREF * (V_{DDA}/ 29.4), for VREF = 0x0 .. 0xF For minimum and maximum V_{IREF} threshold values, see Table 18-3 on page 1087.
	RNG=1	V_{IREF} Low Range: 16 voltage threshold values indexed by VREF = 0x0 .. 0xF Ideal starting voltage (VREF=0): 0 V Ideal step size: V_{DDA}/ 22.12 Ideal V_{IREF} threshold values: V_{IREF} (VREF) = VREF * (V_{DDA}/ 22.12), for VREF = 0x0 .. 0xF For minimum and maximum V_{IREF} threshold values, see Table 18-4 on page 1087.

Note that the values shown in Table 18-2 are the ideal values of the V_{IREF} thresholds. These values actually vary between minimum and maximum values for each threshold step, depending on process and temperature. The minimum and maximum values for each step are given by:

- V_{IREF}(VREF) [Min] = Ideal V_{IREF}(VREF) – (Ideal Step size – 2 mV) / 2
- V_{IREF}(VREF) [Max] = Ideal V_{IREF}(VREF) + (Ideal Step size – 2 mV) / 2

Examples of minimum and maximum V_{IREF} values for V_{DDA} = 3.3V for high and low ranges, are shown inTable 18-3 and Table 18-4. Note that these examples are only valid for V_{DDA} = 3.3V; values scale up and down with V_{DDA}.

Table 18-3. Analog Comparator Voltage Reference Characteristics, V_{DDA} = 3.3V, EN= 1, and RNG = 0

VREF Value	V_{IREF} Min	Ideal V_{IREF}	V_{IREF} Max	Unit
0x0	0.731	0.786	0.841	V
0x1	0.843	0.898	0.953	V
0x2	0.955	1.010	1.065	V
0x3	1.067	1.122	1.178	V
0x4	1.180	1.235	1.290	V
0x5	1.292	1.347	1.402	V
0x6	1.404	1.459	1.514	V
0x7	1.516	1.571	1.627	V
0x8	1.629	1.684	1.739	V
0x9	1.741	1.796	1.851	V
0xA	1.853	1.908	1.963	V
0xB	1.965	2.020	2.076	V
0xC	2.078	2.133	2.188	V
0xD	2.190	2.245	2.300	V
0xE	2.302	2.357	2.412	V
0xF	2.414	2.469	2.525	V

Table 18-4. Analog Comparator Voltage Reference Characteristics, V_{DDA} = 3.3V, EN= 1, and RNG = 1

VREF Value	V_{IREF} Min	Ideal V_{IREF}	V_{IREF} Max	Unit
0x0	0.000	0.000	0.074	V
0x1	0.076	0.149	0.223	V
0x2	0.225	0.298	0.372	V
0x3	0.374	0.448	0.521	V
0x4	0.523	0.597	0.670	V
0x5	0.672	0.746	0.820	V
0x6	0.822	0.895	0.969	V
0x7	0.971	1.044	1.118	V
0x8	1.120	1.193	1.267	V
0x9	1.269	1.343	1.416	V
0xA	1.418	1.492	1.565	V

Table 18-4. Analog Comparator Voltage Reference Characteristics, V_{DDA} = 3.3V, EN= 1, and RNG = 1 *(continued)*

VREF Value	V_{IREF} Min	Ideal V_{IREF}	V_{IREF} Max	Unit
0xB	1.567	1.641	1.715	V
0xC	1.717	1.790	1.864	V
0xD	1.866	1.939	2.013	V
0xE	2.015	2.089	2.162	V
0xF	2.164	2.238	2.311	V

18.4 Initialization and Configuration

The following example shows how to configure an analog comparator to read back its output value from an internal register.

1. Enable the analog comparator clock by writing a value of 0x0000.0001 to the **RCGCACMP** register in the System Control module (see page 331).
2. Enable the clock to the appropriate GPIO modules via the **RCGCGPIO** register (see page 319). To find out which GPIO ports to enable, refer to Table 20-5 on page 1115.
3. In the GPIO module, enable the GPIO port/pin associated with the input signals as GPIO inputs. To determine which GPIO to configure, see Table 20-4 on page 1111.
4. Configure the PMCn fields in the **GPIOPCTL** register to assign the analog comparator output signals to the appropriate pins (see page 620 and Table 20-5 on page 1115).
5. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.
6. Configure the comparator to use the internal voltage reference and to *not* invert the output by writing the **ACCTLn** register with the value of 0x0000.040C.
7. Delay for 10 µs.
8. Read the comparator output value by reading the **ACSTATn** register's OVAL value.

Change the level of the comparator negative input signal C- to see the OVAL value change.

18.5 Register Map

Table 18-5 on page 1088 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 331). There must be a delay of 3 system clocks after the analog comparator module clock is enabled before any analog comparator module registers are accessed.

Table 18-5. Analog Comparators Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ACMIS	RW1C	0x0000.0000	Analog Comparator Masked Interrupt Status	1090
0x004	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	1091

Table 18-5. Analog Comparators Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x008	ACINTEN	RW	0x0000.0000	Analog Comparator Interrupt Enable	1092
0x010	ACREFCTL	RW	0x0000.0000	Analog Comparator Reference Voltage Control	1093
0x020	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	1094
0x024	ACCTL0	RW	0x0000.0000	Analog Comparator Control 0	1095
0x040	ACSTAT1	RO	0x0000.0000	Analog Comparator Status 1	1094
0x044	ACCTL1	RW	0x0000.0000	Analog Comparator Control 1	1095
0xFC0	ACMPPP	RO	0x0003.0003	Analog Comparator Peripheral Properties	1097

18.6 Register Descriptions

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

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000

This register provides a summary of the interrupt status (masked) of the comparators.

Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000
Offset 0x000
Type RW1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:2	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	IN1	RW1C	0	Comparator 1 Masked Interrupt Status Value Description 0 No interrupt has occurred or the interrupt is masked. 1 The `IN1` bits in the **ACRIS** register and the **ACINTEN** registers are set, providing an interrupt to the interrupt controller. This bit is cleared by writing a 1. Clearing this bit also clears the `IN1` bit in the **ACRIS** register.
0	IN0	RW1C	0	Comparator 0 Masked Interrupt Status Value Description 0 No interrupt has occurred or the interrupt is masked. 1 The `IN0` bits in the **ACRIS** register and the **ACINTEN** registers are set, providing an interrupt to the interrupt controller. This bit is cleared by writing a 1. Clearing this bit also clears the `IN0` bit in the **ACRIS** register.

Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004

This register provides a summary of the interrupt status (raw) of the comparators. The bits in this register must be enabled to generate interrupts using the **ACINTEN** register.

Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000
Offset 0x004
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														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
31:2	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	IN1	RO	0	Comparator 1 Interrupt Status Value Description 0 An interrupt has not occurred. 1 Comparator 1 has generated an interruptfor an event as configured by the `ISEN` bit in the **ACCTL1** register. 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 0 An interrupt has not occurred. 1 Comparator 0 has generated an interrupt for an event as configured by the `ISEN` bit in the **ACCTL0** register. This bit is cleared by writing a 1 to the `IN0` bit in the **ACMIS** register.

Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008

This register provides the interrupt enable for the comparators.

Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000
Offset 0x008
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of 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	IN1	RW	0	Comparator 1 Interrupt Enable Value Description 0 A comparator 1 interrupt does not affect the interrupt status. 1 The raw interrupt signal comparator 1 is sent to the interrupt controller.
0	IN0	RW	0	Comparator 0 Interrupt Enable Value Description 0 A comparator 0 interrupt does not affect the interrupt status. 1 The raw interrupt signal comparator 0 is sent to the interrupt controller.

Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010

This register specifies whether the resistor ladder is powered on as well as the range and tap.

Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000
Offset 0x010
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						EN	RNG	reserved				VREF			
Type	RO	RO	RO	RO	RO	RO	RW	RW	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	EN	RW	0	Resistor Ladder Enable 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	RW	0	Resistor Ladder Range Value Description 0 The ideal step size for the internal reference is VDDA / 29.4. 1 The ideal step size for the internal reference is VDDA / 22.12.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	VREF	RW	0x0	Resistor Ladder Voltage Ref The VREF bit field specifies the resistor ladder tap that is passed through an analog multiplexer. The voltage corresponding to the tap position is the internal reference voltage available for comparison. See Table 18-2 on page 1086 for some output reference voltage examples.

Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020

Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040

These registers specify the current output value of the comparator.

Analog Comparator Status n (ACSTATn)

Base 0x4003.C000
Offset 0x020
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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
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	OVAL	RO	0	Comparator Output Value Value Description 0 VIN- > VIN+ 1 VIN- < VIN+ VIN - is the voltage on the `Cn-` pin. VIN+ is the voltage on the `Cn+` pin, the `C0+` pin, or the internal voltage reference (V_{IREF}) as defined by the `ASRCP` bit in the **ACCTL** register.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 7: Analog Comparator Control 0 (ACCTL0), offset 0x024

Register 8: Analog Comparator Control 1 (ACCTL1), offset 0x044

These registers configure the comparator's input and output.

Analog Comparator Control n (ACCTLn)

Base 0x4003.C000
Offset 0x024
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TOEN	ASRCP		reserved	TSLVAL	TSEN		ISLVAL	ISEN		CINV	reserved
Type	RO	RO	RO	RO	RW	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TOEN	RW	0	Trigger Output Enable Value Description 0 ADC events are suppressed and not sent to the ADC. 1 ADC events are sent to the ADC.
10:9	ASRCP	RW	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	RW	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	RW	0x0	Trigger Sense The TSEN field specifies the sense of the comparator output that generates an ADC event. The sense conditioning is as follows: Value Description 0x0 Level sense, see TSLVAL 0x1 Falling edge 0x2 Rising edge 0x3 Either edge
4	ISLVAL	RW	0	Interrupt Sense Level Value 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	RW	0x0	Interrupt Sense The ISEN field specifies the sense of the comparator output that generates an interrupt. The sense conditioning is as follows: Value Description 0x0 Level sense, see ISLVAL 0x1 Falling edge 0x2 Rising edge 0x3 Either edge
1	CINV	RW	0	Comparator Output Invert 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.

Register 9: Analog Comparator Peripheral Properties (ACMPPP), offset 0xFC0

The **ACMPPP** register provides information regarding the properties of the analog comparator module.

Analog Comparator Peripheral Properties (ACMPPP)

Base 0x4003.C000
Offset 0xFC0
Type RO, reset 0x0003.0003

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved														C1O	C0O
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														CMP1	CMP0
Type	RO	RO	RO	RO	RO	RO	RO	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

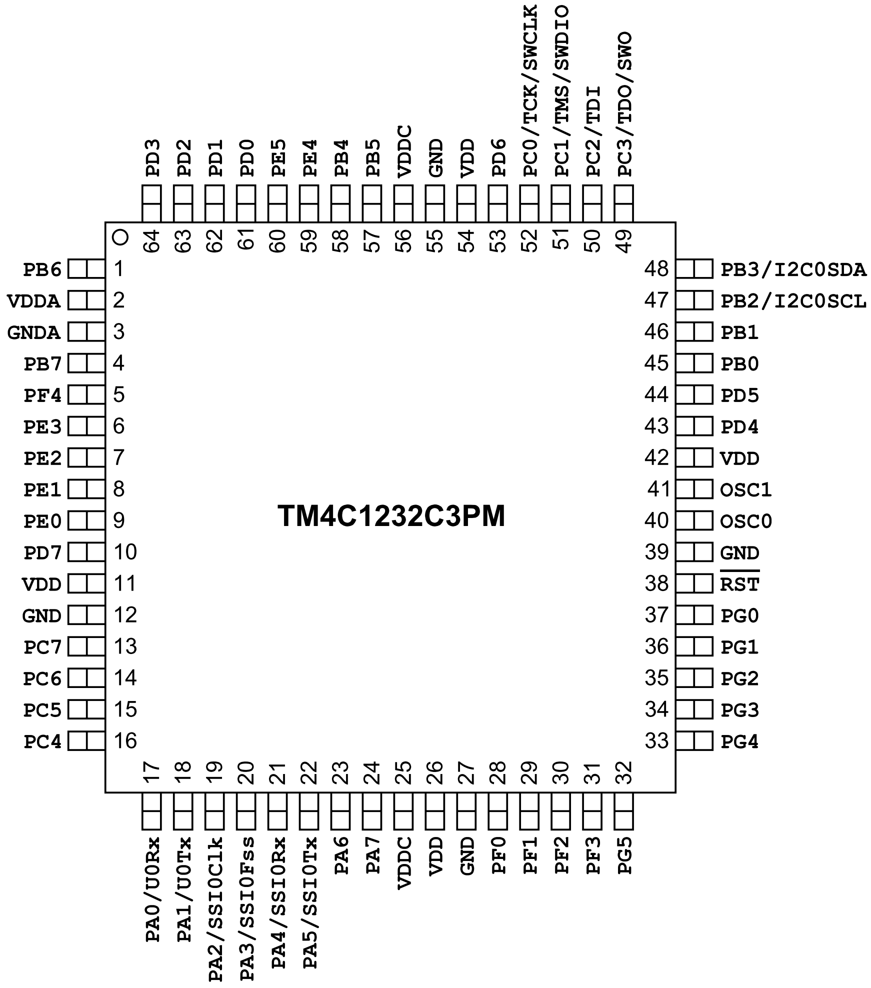
Bit/Field	Name	Type	Reset	Description
31:18	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
17	C1O	RO	0x1	Comparator Output 1 Present Value Description 0 Comparator output 1 is not present. 1 Comparator output 1 is present.
16	C0O	RO	0x1	Comparator Output 0 Present Value Description 0 Comparator output 0 is not present. 1 Comparator output 0 is present.
15:2	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	CMP1	RO	0x1	Comparator 1 Present Value Description 0 Comparator 1 is not present. 1 Comparator 1 is present.
0	CMP0	RO	0x1	Comparator 0 Present Value Description 0 Comparator 0 is not present. 1 Comparator 0 is present.

19 Pin Diagram

The TM4C1232C3PM 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 20-5 on page 1115.

Figure 19-1. 64-Pin LQFP Package Pin Diagram



20 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 619) 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 603) must be set. Further pin muxing options are provided through the `PMCx` bit field in the **GPIOPCTL** register (see page 620), which selects one of several available peripheral functions for that GPIO.

Important: Table 9-1 on page 582 shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ returns these GPIO to their original special consideration state.

Table 20-1. GPIO Pins With Special Considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	I^{21}C0	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO[a]	0	0	0	0	0x0	0
PF[0]	GPIO[a]	0	0	0	0	0x0	0

[a] This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

Table 20-2 on page 1100 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 20-3 on page 1105 lists the signals in alphabetical order by signal name. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed. The "Pin Mux" column indicates the GPIO and the encoding needed in the `PMCx` bit field in the **GPIOPCTL** register.

Table 20-4 on page 1111 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 20-5 on page 1115 lists the GPIO pins and their analog and digital alternate functions. The `AINx` analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding `DEN` bit in the **GPIO Digital Enable (GPIODEN)** register 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+`). These signals are configured by clearing the `DEN` bit in the **GPIO Digital Enable (GPIODEN)** register. The digital signals are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the `PMCx` bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric enoding shown in the table below. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Table 20-6 on page 1118 lists the signals based on number of possible pin assignments. This table can be used to plan how to configure the pins for a particular functionality. Application Note AN01274 Configuring Tiva™ C Series Microcontrollers with Pin Multiplexing provides an overview of the pin muxing implementation, an explanation of how a system designer defines a pin configuration, and examples of the pin configuration process.

Note: All digital inputs are Schmitt triggered.

20.1 Signals by Pin Number

Table 20-2. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
1	PB6	I/O	TTL	GPIO port B bit 6.
	I2C5SCL	I/O	OD	I^{2}C module 5 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	SSI2Rx	I	TTL	SSI module 2 receive.
	T0CCP0	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 0.
2	VDDA	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in Table 21-5 on page 1124, regardless of system implementation.
3	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.
4	PB7	I/O	TTL	GPIO port B bit 7.
	I2C5SDA	I/O	OD	I^{2}C module 5 data.
	SSI2Tx	O	TTL	SSI module 2 transmit.
	T0CCP1	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 1.
5	PF4	I/O	TTL	GPIO port F bit 4.
	T2CCP0	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 0.
6	PE3	I/O	TTL	GPIO port E bit 3.
	AIN0	I	Analog	Analog-to-digital converter input 0.
7	PE2	I/O	TTL	GPIO port E bit 2.
	AIN1	I	Analog	Analog-to-digital converter input 1.
8	PE1	I/O	TTL	GPIO port E bit 1.
	AIN2	I	Analog	Analog-to-digital converter input 2.
	U7Tx	O	TTL	UART module 7 transmit.
9	PE0	I/O	TTL	GPIO port E bit 0.
	AIN3	I	Analog	Analog-to-digital converter input 3.
	U7Rx	I	TTL	UART module 7 receive.
10	PD7	I/O	TTL	GPIO port D bit 7.
	NMI	I	TTL	Non-maskable interrupt.
	U2Tx	O	TTL	UART module 2 transmit.
	WT5CCP1	I/O	TTL	32/64-Bit Wide Timer 5 Capture/Compare/PWM 1.
11	VDD	-	Power	Positive supply for I/O and some logic.

Table 20-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
12	GND	-	Power	Ground reference for logic and I/O pins.
13	PC7	I/O	TTL	GPIO port C bit 7.
	C0-	I	Analog	Analog comparator 0 negative input.
	U3Tx	O	TTL	UART module 3 transmit.
	WT1CCP1	I/O	TTL	32/64-Bit Wide Timer 1 Capture/Compare/PWM 1.
14	PC6	I/O	TTL	GPIO port C bit 6.
	C0+	I	Analog	Analog comparator 0 positive input.
	U3Rx	I	TTL	UART module 3 receive.
	WT1CCP0	I/O	TTL	32/64-Bit Wide Timer 1 Capture/Compare/PWM 0.
15	PC5	I/O	TTL	GPIO port C bit 5.
	C1+	I	Analog	Analog comparator 1 positive input.
	U1CTS	I	TTL	UART module 1 Clear To Send modem flow control input signal.
	U1Tx	O	TTL	UART module 1 transmit.
	U4Tx	O	TTL	UART module 4 transmit.
	WT0CCP1	I/O	TTL	32/64-Bit Wide Timer 0 Capture/Compare/PWM 1.
16	PC4	I/O	TTL	GPIO port C bit 4.
	C1-	I	Analog	Analog comparator 1 negative input.
	U1RTS	O	TTL	UART module 1 Request to Send modem flow control output line.
	U1Rx	I	TTL	UART module 1 receive.
	U4Rx	I	TTL	UART module 4 receive.
	WT0CCP0	I/O	TTL	32/64-Bit Wide Timer 0 Capture/Compare/PWM 0.
17	PA0	I/O	TTL	GPIO port A bit 0.
	U0Rx	I	TTL	UART module 0 receive.
18	PA1	I/O	TTL	GPIO port A bit 1.
	U0Tx	O	TTL	UART module 0 transmit.
19	PA2	I/O	TTL	GPIO port A bit 2.
	SSI0Clk	I/O	TTL	SSI module 0 clock
20	PA3	I/O	TTL	GPIO port A bit 3.
	SSI0Fss	I/O	TTL	SSI module 0 frame signal
21	PA4	I/O	TTL	GPIO port A bit 4.
	SSI0Rx	I	TTL	SSI module 0 receive
22	PA5	I/O	TTL	GPIO port A bit 5.
	SSI0Tx	O	TTL	SSI module 0 transmit
23	PA6	I/O	TTL	GPIO port A bit 6.
	I2C1SCL	I/O	OD	I^{2}C module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
24	PA7	I/O	TTL	GPIO port A bit 7.
	I2C1SDA	I/O	OD	I^{2}C module 1 data.
25	VDDC	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.2 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to each other and an external capacitor as specified in Table 21-12 on page 1137 .

Table 20-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
26	VDD	-	Power	Positive supply for I/O and some logic.
27	GND	-	Power	Ground reference for logic and I/O pins.
28	PF0	I/O	TTL	GPIO port F bit 0.
	C0o	O	TTL	Analog comparator 0 output.
	CAN0Rx	I	TTL	CAN module 0 receive.
	NMI	I	TTL	Non-maskable interrupt.
	SSI1Rx	I	TTL	SSI module 1 receive.
	T0CCP0	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 0.
	U1RTS	O	TTL	UART module 1 Request to Send modem flow control output line.
29	PF1	I/O	TTL	GPIO port F bit 1.
	C1o	O	TTL	Analog comparator 1 output.
	SSI1Tx	O	TTL	SSI module 1 transmit.
	T0CCP1	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 1.
	TRD1	O	TTL	Trace data 1.
	U1CTS	I	TTL	UART module 1 Clear To Send modem flow control input signal.
30	PF2	I/O	TTL	GPIO port F bit 2.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	T1CCP0	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 0.
	TRD0	O	TTL	Trace data 0.
31	PF3	I/O	TTL	GPIO port F bit 3.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	SSI1Fss	I/O	TTL	SSI module 1 frame signal.
	T1CCP1	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 1.
	TRCLK	O	TTL	Trace clock.
32	PG5	I/O	TTL	GPIO port G bit 5.
	I2C1SDA	I/O	OD	I^{2}C module 1 data.
	U2Tx	O	TTL	UART module 2 transmit.
	WT0CCP1	I/O	TTL	32/64-Bit Wide Timer 0 Capture/Compare/PWM 1.
33	PG4	I/O	TTL	GPIO port G bit 4.
	I2C1SCL	I/O	OD	I^{2}C module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	U2Rx	I	TTL	UART module 2 receive.
	WT0CCP0	I/O	TTL	32/64-Bit Wide Timer 0 Capture/Compare/PWM 0.
34	PG3	I/O	TTL	GPIO port G bit 3.
	I2C4SDA	I/O	OD	I^{2}C module 4 data.
	T5CCP1	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 1.
35	PG2	I/O	TTL	GPIO port G bit 2.
	I2C4SCL	I/O	OD	I^{2}C module 4 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	T5CCP0	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 0.

Table 20-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
36	PG1	I/O	TTL	GPIO port G bit 1.
	I2C3SDA	I/O	OD	I^{2}C module 3 data.
	T4CCP1	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 1.
37	PG0	I/O	TTL	GPIO port G bit 0.
	I2C3SCL	I/O	OD	I^{2}C module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	T4CCP0	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 0.
38	$\overline{\text{RST}}$	I	TTL	System reset input.
39	GND	-	Power	Ground reference for logic and I/O pins.
40	OSC0	I	Analog	Main oscillator crystal input or an external clock reference input.
41	OSC1	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
42	VDD	-	Power	Positive supply for I/O and some logic.
43	PD4	I/O	TTL	GPIO port D bit 4. This pin is not 5-V tolerant.
	U6Rx	I	TTL	UART module 6 receive.
	USB0DM	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
	WT4CCP0	I/O	TTL	32/64-Bit Wide Timer 4 Capture/Compare/PWM 0.
44	PD5	I/O	TTL	GPIO port D bit 5. This pin is not 5-V tolerant.
	U6Tx	O	TTL	UART module 6 transmit.
	USB0DP	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.
	WT4CCP1	I/O	TTL	32/64-Bit Wide Timer 4 Capture/Compare/PWM 1.
45	PB0	I/O	TTL	GPIO port B bit 0. This pin is not 5-V tolerant.
	T2CCP0	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 0.
	U1Rx	I	TTL	UART module 1 receive.
46	PB1	I/O	TTL	GPIO port B bit 1. This pin is not 5-V tolerant.
	T2CCP1	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 1.
	U1Tx	O	TTL	UART module 1 transmit.
47	PB2	I/O	TTL	GPIO port B bit 2.
	I2C0SCL	I/O	OD	I^{2}C module 0 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	T3CCP0	I/O	TTL	16/32-Bit Timer 3 Capture/Compare/PWM 0.
48	PB3	I/O	TTL	GPIO port B bit 3.
	I2C0SDA	I/O	OD	I^{2}C module 0 data.
	T3CCP1	I/O	TTL	16/32-Bit Timer 3 Capture/Compare/PWM 1.
49	PC3	I/O	TTL	GPIO port C bit 3.
	SWO	O	TTL	JTAG TDO and SWO.
	T5CCP1	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 1.
	TDO	O	TTL	JTAG TDO and SWO.
50	PC2	I/O	TTL	GPIO port C bit 2.
	T5CCP0	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 0.
	TDI	I	TTL	JTAG TDI.

Table 20-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
51	PC1	I/O	TTL	GPIO port C bit 1.
	SWDIO	I/O	TTL	JTAG TMS and SWDIO.
	T4CCP1	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 1.
	TMS	I	TTL	JTAG TMS and SWDIO.
52	PC0	I/O	TTL	GPIO port C bit 0.
	SWCLK	I	TTL	JTAG/SWD CLK.
	T4CCP0	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 0.
	TCK	I	TTL	JTAG/SWD CLK.
53	PD6	I/O	TTL	GPIO port D bit 6.
	U2Rx	I	TTL	UART module 2 receive.
	WT5CCP0	I/O	TTL	32/64-Bit Wide Timer 5 Capture/Compare/PWM 0.
54	VDD	-	Power	Positive supply for I/O and some logic.
55	GND	-	Power	Ground reference for logic and I/O pins.
56	VDDC	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.2 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to each other and an external capacitor as specified in Table 21-12 on page 1137 .
57	PB5	I/O	TTL	GPIO port B bit 5.
	AIN11	I	Analog	Analog-to-digital converter input 11.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	SSI2Fss	I/O	TTL	SSI module 2 frame signal.
	T1CCP1	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 1.
58	PB4	I/O	TTL	GPIO port B bit 4.
	AIN10	I	Analog	Analog-to-digital converter input 10.
	CAN0Rx	I	TTL	CAN module 0 receive.
	SSI2Clk	I/O	TTL	SSI module 2 clock.
	T1CCP0	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 0.
59	PE4	I/O	TTL	GPIO port E bit 4.
	AIN9	I	Analog	Analog-to-digital converter input 9.
	CAN0Rx	I	TTL	CAN module 0 receive.
	I2C2SCL	I/O	OD	I^{2}C module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	U5Rx	I	TTL	UART module 5 receive.
60	PE5	I/O	TTL	GPIO port E bit 5.
	AIN8	I	Analog	Analog-to-digital converter input 8.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	I2C2SDA	I/O	OD	I^{2}C module 2 data.
	U5Tx	O	TTL	UART module 5 transmit.

Table 20-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
61	PD0	I/O	TTL	GPIO port D bit 0.
	AIN7	I	Analog	Analog-to-digital converter input 7.
	I2C3SCL	I/O	OD	I^{2}C module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	SSI3Clk	I/O	TTL	SSI module 3 clock.
	WT2CCP0	I/O	TTL	32/64-Bit Wide Timer 2 Capture/Compare/PWM 0.
62	PD1	I/O	TTL	GPIO port D bit 1.
	AIN6	I	Analog	Analog-to-digital converter input 6.
	I2C3SDA	I/O	OD	I^{2}C module 3 data.
	SSI1Fss	I/O	TTL	SSI module 1 frame signal.
	SSI3Fss	I/O	TTL	SSI module 3 frame signal.
	WT2CCP1	I/O	TTL	32/64-Bit Wide Timer 2 Capture/Compare/PWM 1.
63	PD2	I/O	TTL	GPIO port D bit 2.
	AIN5	I	Analog	Analog-to-digital converter input 5.
	SSI1Rx	I	TTL	SSI module 1 receive.
	SSI3Rx	I	TTL	SSI module 3 receive.
	WT3CCP0	I/O	TTL	32/64-Bit Wide Timer 3 Capture/Compare/PWM 0.
64	PD3	I/O	TTL	GPIO port D bit 3.
	AIN4	I	Analog	Analog-to-digital converter input 4.
	SSI1Tx	O	TTL	SSI module 1 transmit.
	SSI3Tx	O	TTL	SSI module 3 transmit.
	WT3CCP1	I/O	TTL	32/64-Bit Wide Timer 3 Capture/Compare/PWM 1.

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

20.2 Signals by Signal Name

Table 20-3. Signals by Signal Name

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN0	6	PE3	I	Analog	Analog-to-digital converter input 0.
AIN1	7	PE2	I	Analog	Analog-to-digital converter input 1.
AIN2	8	PE1	I	Analog	Analog-to-digital converter input 2.
AIN3	9	PE0	I	Analog	Analog-to-digital converter input 3.
AIN4	64	PD3	I	Analog	Analog-to-digital converter input 4.
AIN5	63	PD2	I	Analog	Analog-to-digital converter input 5.
AIN6	62	PD1	I	Analog	Analog-to-digital converter input 6.
AIN7	61	PD0	I	Analog	Analog-to-digital converter input 7.
AIN8	60	PE5	I	Analog	Analog-to-digital converter input 8.
AIN9	59	PE4	I	Analog	Analog-to-digital converter input 9.
AIN10	58	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	57	PB5	I	Analog	Analog-to-digital converter input 11.

Table 20-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
C0+	14	PC6	I	Analog	Analog comparator 0 positive input.
C0-	13	PC7	I	Analog	Analog comparator 0 negative input.
C0o	28	PF0 (9)	O	TTL	Analog comparator 0 output.
C1+	15	PC5	I	Analog	Analog comparator 1 positive input.
C1-	16	PC4	I	Analog	Analog comparator 1 negative input.
C1o	29	PF1 (9)	O	TTL	Analog comparator 1 output.
CAN0Rx	28 58 59	PF0 (3) PB4 (8) PE4 (8)	I	TTL	CAN module 0 receive.
CAN0Tx	31 57 60	PF3 (3) PB5 (8) PE5 (8)	O	TTL	CAN module 0 transmit.
GND	12 27 39 55	fixed	-	Power	Ground reference for logic and I/O pins.
GNDA	3	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.
I2C0SCL	47	PB2 (3)	I/O	OD	I^{2}C module 0 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C0SDA	48	PB3 (3)	I/O	OD	I^{2}C module 0 data.
I2C1SCL	23 33	PA6 (3) PG4 (3)	I/O	OD	I^{2}C module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C1SDA	24 32	PA7 (3) PG5 (3)	I/O	OD	I^{2}C module 1 data.
I2C2SCL	59	PE4 (3)	I/O	OD	I^{2}C module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C2SDA	60	PE5 (3)	I/O	OD	I^{2}C module 2 data.
I2C3SCL	37 61	PG0 (3) PD0 (3)	I/O	OD	I^{2}C module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C3SDA	36 62	PG1 (3) PD1 (3)	I/O	OD	I^{2}C module 3 data.
I2C4SCL	35	PG2 (3)	I/O	OD	I^{2}C module 4 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C4SDA	34	PG3 (3)	I/O	OD	I^{2}C module 4 data.
I2C5SCL	1	PB6 (3)	I/O	OD	I^{2}C module 5 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C5SDA	4	PB7 (3)	I/O	OD	I^{2}C module 5 data.
NMI	10 28	PD7 (8) PF0 (8)	I	TTL	Non-maskable interrupt.

Table 20-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
OSC0	40	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	41	fixed	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
PA0	17	-	I/O	TTL	GPIO port A bit 0.
PA1	18	-	I/O	TTL	GPIO port A bit 1.
PA2	19	-	I/O	TTL	GPIO port A bit 2.
PA3	20	-	I/O	TTL	GPIO port A bit 3.
PA4	21	-	I/O	TTL	GPIO port A bit 4.
PA5	22	-	I/O	TTL	GPIO port A bit 5.
PA6	23	-	I/O	TTL	GPIO port A bit 6.
PA7	24	-	I/O	TTL	GPIO port A bit 7.
PB0	45	-	I/O	TTL	GPIO port B bit 0. This pin is not 5-V tolerant.
PB1	46	-	I/O	TTL	GPIO port B bit 1. This pin is not 5-V tolerant.
PB2	47	-	I/O	TTL	GPIO port B bit 2.
PB3	48	-	I/O	TTL	GPIO port B bit 3.
PB4	58	-	I/O	TTL	GPIO port B bit 4.
PB5	57	-	I/O	TTL	GPIO port B bit 5.
PB6	1	-	I/O	TTL	GPIO port B bit 6.
PB7	4	-	I/O	TTL	GPIO port B bit 7.
PC0	52	-	I/O	TTL	GPIO port C bit 0.
PC1	51	-	I/O	TTL	GPIO port C bit 1.
PC2	50	-	I/O	TTL	GPIO port C bit 2.
PC3	49	-	I/O	TTL	GPIO port C bit 3.
PC4	16	-	I/O	TTL	GPIO port C bit 4.
PC5	15	-	I/O	TTL	GPIO port C bit 5.
PC6	14	-	I/O	TTL	GPIO port C bit 6.
PC7	13	-	I/O	TTL	GPIO port C bit 7.
PD0	61	-	I/O	TTL	GPIO port D bit 0.
PD1	62	-	I/O	TTL	GPIO port D bit 1.
PD2	63	-	I/O	TTL	GPIO port D bit 2.
PD3	64	-	I/O	TTL	GPIO port D bit 3.
PD4	43	-	I/O	TTL	GPIO port D bit 4. This pin is not 5-V tolerant.
PD5	44	-	I/O	TTL	GPIO port D bit 5. This pin is not 5-V tolerant.
PD6	53	-	I/O	TTL	GPIO port D bit 6.
PD7	10	-	I/O	TTL	GPIO port D bit 7.
PE0	9	-	I/O	TTL	GPIO port E bit 0.
PE1	8	-	I/O	TTL	GPIO port E bit 1.
PE2	7	-	I/O	TTL	GPIO port E bit 2.
PE3	6	-	I/O	TTL	GPIO port E bit 3.
PE4	59	-	I/O	TTL	GPIO port E bit 4.

Table 20-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
PE5	60	-	I/O	TTL	GPIO port E bit 5.
PF0	28	-	I/O	TTL	GPIO port F bit 0.
PF1	29	-	I/O	TTL	GPIO port F bit 1.
PF2	30	-	I/O	TTL	GPIO port F bit 2.
PF3	31	-	I/O	TTL	GPIO port F bit 3.
PF4	5	-	I/O	TTL	GPIO port F bit 4.
PG0	37	-	I/O	TTL	GPIO port G bit 0.
PG1	36	-	I/O	TTL	GPIO port G bit 1.
PG2	35	-	I/O	TTL	GPIO port G bit 2.
PG3	34	-	I/O	TTL	GPIO port G bit 3.
PG4	33	-	I/O	TTL	GPIO port G bit 4.
PG5	32	-	I/O	TTL	GPIO port G bit 5.
$\overline{\text{RST}}$	38	fixed	I	TTL	System reset input.
SSI0Clk	19	PA2 (2)	I/O	TTL	SSI module 0 clock
SSI0Fss	20	PA3 (2)	I/O	TTL	SSI module 0 frame signal
SSI0Rx	21	PA4 (2)	I	TTL	SSI module 0 receive
SSI0Tx	22	PA5 (2)	O	TTL	SSI module 0 transmit
SSI1Clk	30 61	PF2 (2) PD0 (2)	I/O	TTL	SSI module 1 clock.
SSI1Fss	31 62	PF3 (2) PD1 (2)	I/O	TTL	SSI module 1 frame signal.
SSI1Rx	28 63	PF0 (2) PD2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	29 64	PF1 (2) PD3 (2)	O	TTL	SSI module 1 transmit.
SSI2Clk	58	PB4 (2)	I/O	TTL	SSI module 2 clock.
SSI2Fss	57	PB5 (2)	I/O	TTL	SSI module 2 frame signal.
SSI2Rx	1	PB6 (2)	I	TTL	SSI module 2 receive.
SSI2Tx	4	PB7 (2)	O	TTL	SSI module 2 transmit.
SSI3Clk	61	PD0 (1)	I/O	TTL	SSI module 3 clock.
SSI3Fss	62	PD1 (1)	I/O	TTL	SSI module 3 frame signal.
SSI3Rx	63	PD2 (1)	I	TTL	SSI module 3 receive.
SSI3Tx	64	PD3 (1)	O	TTL	SSI module 3 transmit.
SWCLK	52	PC0 (1)	I	TTL	JTAG/SWD CLK.
SWDIO	51	PC1 (1)	I/O	TTL	JTAG TMS and SWDIO.
SWO	49	PC3 (1)	O	TTL	JTAG TDO and SWO.
T0CCP0	1 28	PB6 (7) PF0 (7)	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 0.
T0CCP1	4 29	PB7 (7) PF1 (7)	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 1.
T1CCP0	30 58	PF2 (7) PB4 (7)	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 0.

Table 20-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
T1CCP1	31 57	PF3 (7) PB5 (7)	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 1.
T2CCP0	5 45	PF4 (7) PB0 (7)	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 0.
T2CCP1	46	PB1 (7)	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 1.
T3CCP0	47	PB2 (7)	I/O	TTL	16/32-Bit Timer 3 Capture/Compare/PWM 0.
T3CCP1	48	PB3 (7)	I/O	TTL	16/32-Bit Timer 3 Capture/Compare/PWM 1.
T4CCP0	37 52	PG0 (7) PC0 (7)	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 0.
T4CCP1	36 51	PG1 (7) PC1 (7)	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 1.
T5CCP0	35 50	PG2 (7) PC2 (7)	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 0.
T5CCP1	34 49	PG3 (7) PC3 (7)	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 1.
TCK	52	PC0 (1)	I	TTL	JTAG/SWD CLK.
TDI	50	PC2 (1)	I	TTL	JTAG TDI.
TDO	49	PC3 (1)	O	TTL	JTAG TDO and SWO.
TMS	51	PC1 (1)	I	TTL	JTAG TMS and SWDIO.
TRCLK	31	PF3 (14)	O	TTL	Trace clock.
TRD0	30	PF2 (14)	O	TTL	Trace data 0.
TRD1	29	PF1 (14)	O	TTL	Trace data 1.
U0Rx	17	PA0 (1)	I	TTL	UART module 0 receive.
U0Tx	18	PA1 (1)	O	TTL	UART module 0 transmit.
U1CTS	15 29	PC5 (8) PF1 (1)	I	TTL	UART module 1 Clear To Send modem flow control input signal.
U1RTS	16 28	PC4 (8) PF0 (1)	O	TTL	UART module 1 Request to Send modem flow control output line.
U1Rx	16 45	PC4 (2) PB0 (1)	I	TTL	UART module 1 receive.
U1Tx	15 46	PC5 (2) PB1 (1)	O	TTL	UART module 1 transmit.
U2Rx	33 53	PG4 (1) PD6 (1)	I	TTL	UART module 2 receive.
U2Tx	10 32	PD7 (1) PG5 (1)	O	TTL	UART module 2 transmit.
U3Rx	14	PC6 (1)	I	TTL	UART module 3 receive.
U3Tx	13	PC7 (1)	O	TTL	UART module 3 transmit.
U4Rx	16	PC4 (1)	I	TTL	UART module 4 receive.
U4Tx	15	PC5 (1)	O	TTL	UART module 4 transmit.
U5Rx	59	PE4 (1)	I	TTL	UART module 5 receive.
U5Tx	60	PE5 (1)	O	TTL	UART module 5 transmit.
U6Rx	43	PD4 (1)	I	TTL	UART module 6 receive.
U6Tx	44	PD5 (1)	O	TTL	UART module 6 transmit.

Table 20-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
U7Rx	9	PE0 (1)	I	TTL	UART module 7 receive.
U7Tx	8	PE1 (1)	O	TTL	UART module 7 transmit.
USB0DM	43	PD4	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
USB0DP	44	PD5	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.
VDD	11 26 42 54	fixed	-	Power	Positive supply for I/O and some logic.
VDDA	2	fixed	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in Table 21-5 on page 1124, regardless of system implementation.
VDDC	25 56	fixed	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.2 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to each other and an external capacitor as specified in Table 21-12 on page 1137 .
WT0CCP0	16 33	PC4 (7) PG4 (7)	I/O	TTL	32/64-Bit Wide Timer 0 Capture/Compare/PWM 0.
WT0CCP1	15 32	PC5 (7) PG5 (7)	I/O	TTL	32/64-Bit Wide Timer 0 Capture/Compare/PWM 1.
WT1CCP0	14	PC6 (7)	I/O	TTL	32/64-Bit Wide Timer 1 Capture/Compare/PWM 0.
WT1CCP1	13	PC7 (7)	I/O	TTL	32/64-Bit Wide Timer 1 Capture/Compare/PWM 1.
WT2CCP0	61	PD0 (7)	I/O	TTL	32/64-Bit Wide Timer 2 Capture/Compare/PWM 0.
WT2CCP1	62	PD1 (7)	I/O	TTL	32/64-Bit Wide Timer 2 Capture/Compare/PWM 1.
WT3CCP0	63	PD2 (7)	I/O	TTL	32/64-Bit Wide Timer 3 Capture/Compare/PWM 0.
WT3CCP1	64	PD3 (7)	I/O	TTL	32/64-Bit Wide Timer 3 Capture/Compare/PWM 1.
WT4CCP0	43	PD4 (7)	I/O	TTL	32/64-Bit Wide Timer 4 Capture/Compare/PWM 0.
WT4CCP1	44	PD5 (7)	I/O	TTL	32/64-Bit Wide Timer 4 Capture/Compare/PWM 1.
WT5CCP0	53	PD6 (7)	I/O	TTL	32/64-Bit Wide Timer 5 Capture/Compare/PWM 0.
WT5CCP1	10	PD7 (7)	I/O	TTL	32/64-Bit Wide Timer 5 Capture/Compare/PWM 1.

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

20.3 Signals by Function, Except for GPIO

Table 20-4. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC	AIN0	6	I	Analog	Analog-to-digital converter input 0.
	AIN1	7	I	Analog	Analog-to-digital converter input 1.
	AIN2	8	I	Analog	Analog-to-digital converter input 2.
	AIN3	9	I	Analog	Analog-to-digital converter input 3.
	AIN4	64	I	Analog	Analog-to-digital converter input 4.
	AIN5	63	I	Analog	Analog-to-digital converter input 5.
	AIN6	62	I	Analog	Analog-to-digital converter input 6.
	AIN7	61	I	Analog	Analog-to-digital converter input 7.
	AIN8	60	I	Analog	Analog-to-digital converter input 8.
	AIN9	59	I	Analog	Analog-to-digital converter input 9.
	AIN10	58	I	Analog	Analog-to-digital converter input 10.
	AIN11	57	I	Analog	Analog-to-digital converter input 11.
Analog Comparators	C0+	14	I	Analog	Analog comparator 0 positive input.
	C0-	13	I	Analog	Analog comparator 0 negative input.
	C0o	28	O	TTL	Analog comparator 0 output.
	C1+	15	I	Analog	Analog comparator 1 positive input.
	C1-	16	I	Analog	Analog comparator 1 negative input.
	C1o	29	O	TTL	Analog comparator 1 output.
Controller Area Network	CAN0Rx	28 58 59	I	TTL	CAN module 0 receive.
	CAN0Tx	31 57 60	O	TTL	CAN module 0 transmit.
Core	TRCLK	31	O	TTL	Trace clock.
	TRD0	30	O	TTL	Trace data 0.
	TRD1	29	O	TTL	Trace data 1.

Table 20-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
General-Purpose Timers	T0CCP0	1 28	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 0.
	T0CCP1	4 29	I/O	TTL	16/32-Bit Timer 0 Capture/Compare/PWM 1.
	T1CCP0	30 58	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 0.
	T1CCP1	31 57	I/O	TTL	16/32-Bit Timer 1 Capture/Compare/PWM 1.
	T2CCP0	5 45	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 0.
	T2CCP1	46	I/O	TTL	16/32-Bit Timer 2 Capture/Compare/PWM 1.
	T3CCP0	47	I/O	TTL	16/32-Bit Timer 3 Capture/Compare/PWM 0.
	T3CCP1	48	I/O	TTL	16/32-Bit Timer 3 Capture/Compare/PWM 1.
	T4CCP0	37 52	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 0.
	T4CCP1	36 51	I/O	TTL	16/32-Bit Timer 4 Capture/Compare/PWM 1.
	T5CCP0	35 50	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 0.
	T5CCP1	34 49	I/O	TTL	16/32-Bit Timer 5 Capture/Compare/PWM 1.
	WT0CCP0	16 33	I/O	TTL	32/64-Bit Wide Timer 0 Capture/Compare/PWM 0.
	WT0CCP1	15 32	I/O	TTL	32/64-Bit Wide Timer 0 Capture/Compare/PWM 1.
	WT1CCP0	14	I/O	TTL	32/64-Bit Wide Timer 1 Capture/Compare/PWM 0.
	WT1CCP1	13	I/O	TTL	32/64-Bit Wide Timer 1 Capture/Compare/PWM 1.
	WT2CCP0	61	I/O	TTL	32/64-Bit Wide Timer 2 Capture/Compare/PWM 0.
	WT2CCP1	62	I/O	TTL	32/64-Bit Wide Timer 2 Capture/Compare/PWM 1.
	WT3CCP0	63	I/O	TTL	32/64-Bit Wide Timer 3 Capture/Compare/PWM 0.
	WT3CCP1	64	I/O	TTL	32/64-Bit Wide Timer 3 Capture/Compare/PWM 1.
	WT4CCP0	43	I/O	TTL	32/64-Bit Wide Timer 4 Capture/Compare/PWM 0.
	WT4CCP1	44	I/O	TTL	32/64-Bit Wide Timer 4 Capture/Compare/PWM 1.
	WT5CCP0	53	I/O	TTL	32/64-Bit Wide Timer 5 Capture/Compare/PWM 0.
	WT5CCP1	10	I/O	TTL	32/64-Bit Wide Timer 5 Capture/Compare/PWM 1.

Table 20-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
I2C	I2C0SCL	47	I/O	OD	I^{2}C module 0 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	I2C0SDA	48	I/O	OD	I^{2}C module 0 data.
	I2C1SCL	23 33	I/O	OD	I^{2}C module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	I2C1SDA	24 32	I/O	OD	I^{2}C module 1 data.
	I2C2SCL	59	I/O	OD	I^{2}C module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	I2C2SDA	60	I/O	OD	I^{2}C module 2 data.
	I2C3SCL	37 61	I/O	OD	I^{2}C module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	I2C3SDA	36 62	I/O	OD	I^{2}C module 3 data.
	I2C4SCL	35	I/O	OD	I^{2}C module 4 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	I2C4SDA	34	I/O	OD	I^{2}C module 4 data.
	I2C5SCL	1	I/O	OD	I^{2}C module 5 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
	I2C5SDA	4	I/O	OD	I^{2}C module 5 data.
JTAG/SWD/SWO	SWCLK	52	I	TTL	JTAG/SWD CLK.
	SWDIO	51	I/O	TTL	JTAG TMS and SWDIO.
	SWO	49	O	TTL	JTAG TDO and SWO.
	TCK	52	I	TTL	JTAG/SWD CLK.
	TDI	50	I	TTL	JTAG TDI.
	TDO	49	O	TTL	JTAG TDO and SWO.
	TMS	51	I	TTL	JTAG TMS and SWDIO.

Table 20-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Power	GND	12 27 39 55	-	Power	Ground reference for logic and I/O pins.
	GNDA	3	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	VDD	11 26 42 54	-	Power	Positive supply for I/O and some logic.
	VDDA	2	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in Table 21-5 on page 1124, regardless of system implementation.
	VDDC	25 56	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.2 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to each other and an external capacitor as specified in Table 21-12 on page 1137 .
SSI	SSI0Clk	19	I/O	TTL	SSI module 0 clock
	SSI0Fss	20	I/O	TTL	SSI module 0 frame signal
	SSI0Rx	21	I	TTL	SSI module 0 receive
	SSI0Tx	22	O	TTL	SSI module 0 transmit
	SSI1Clk	30 61	I/O	TTL	SSI module 1 clock.
	SSI1Fss	31 62	I/O	TTL	SSI module 1 frame signal.
	SSI1Rx	28 63	I	TTL	SSI module 1 receive.
	SSI1Tx	29 64	O	TTL	SSI module 1 transmit.
	SSI2Clk	58	I/O	TTL	SSI module 2 clock.
	SSI2Fss	57	I/O	TTL	SSI module 2 frame signal.
	SSI2Rx	1	I	TTL	SSI module 2 receive.
	SSI2Tx	4	O	TTL	SSI module 2 transmit.
	SSI3Clk	61	I/O	TTL	SSI module 3 clock.
	SSI3Fss	62	I/O	TTL	SSI module 3 frame signal.
	SSI3Rx	63	I	TTL	SSI module 3 receive.
	SSI3Tx	64	O	TTL	SSI module 3 transmit.

Table 20-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
System Control & Clocks	NMI	10 28	I	TTL	Non-maskable interrupt.
	OSC0	40	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	41	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
	$\overline{\text{RST}}$	38	I	TTL	System reset input.
UART	U0Rx	17	I	TTL	UART module 0 receive.
	U0Tx	18	O	TTL	UART module 0 transmit.
	U1CTS	15 29	I	TTL	UART module 1 Clear To Send modem flow control input signal.
	U1RTS	16 28	O	TTL	UART module 1 Request to Send modem flow control output line.
	U1Rx	16 45	I	TTL	UART module 1 receive.
	U1Tx	15 46	O	TTL	UART module 1 transmit.
	U2Rx	33 53	I	TTL	UART module 2 receive.
	U2Tx	10 32	O	TTL	UART module 2 transmit.
	U3Rx	14	I	TTL	UART module 3 receive.
	U3Tx	13	O	TTL	UART module 3 transmit.
	U4Rx	16	I	TTL	UART module 4 receive.
	U4Tx	15	O	TTL	UART module 4 transmit.
	U5Rx	59	I	TTL	UART module 5 receive.
	U5Tx	60	O	TTL	UART module 5 transmit.
	U6Rx	43	I	TTL	UART module 6 receive.
	U6Tx	44	O	TTL	UART module 6 transmit.
	U7Rx	9	I	TTL	UART module 7 receive.
	U7Tx	8	O	TTL	UART module 7 transmit.
USB	USB0DM	43	I/O	Analog	Bidirectional differential data pin (D- per USB specification) for USB0.
	USB0DP	44	I/O	Analog	Bidirectional differential data pin (D+ per USB specification) for USB0.

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

20.4 GPIO Pins and Alternate Functions

Table 20-5. 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	14	15
PA0	17	-	U0Rx	-	-	-	-	-	-	-	-	-	-
PA1	18	-	U0Tx	-	-	-	-	-	-	-	-	-	-
PA2	19	-	-	SSI0Clk	-	-	-	-	-	-	-	-	-

Table 20-5. GPIO Pins and Alternate Functions *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PA3	20	-	-	SSI0Fss	-	-	-	-	-	-	-	-	-
PA4	21	-	-	SSI0Rx	-	-	-	-	-	-	-	-	-
PA5	22	-	-	SSI0Tx	-	-	-	-	-	-	-	-	-
PA6	23	-	-	-	I2C1SCL	-	-	-	-	-	-	-	-
PA7	24	-	-	-	I2C1SDA	-	-	-	-	-	-	-	-
PB0	45	-	U1Rx	-	-	-	-	-	T2CCP0	-	-	-	-
PB1	46	-	U1Tx	-	-	-	-	-	T2CCP1	-	-	-	-
PB2	47	-	-	-	I2C0SCL	-	-	-	T3CCP0	-	-	-	-
PB3	48	-	-	-	I2C0SDA	-	-	-	T3CCP1	-	-	-	-
PB4	58	AIN10	-	SSI2Clk	-	-	-	-	T1CCP0	CAN0Rx	-	-	-
PB5	57	AIN11	-	SSI2Fss	-	-	-	-	T1CCP1	CAN0Tx	-	-	-
PB6	1	-	-	SSI2Rx	I2C5SCL	-	-	-	T0CCP0	-	-	-	-
PB7	4	-	-	SSI2Tx	I2C5SDA	-	-	-	T0CCP1	-	-	-	-
PC0	52	-	TCK SWCLK	-	-	-	-	-	T4CCP0	-	-	-	-
PC1	51	-	TMS SWDIO	-	-	-	-	-	T4CCP1	-	-	-	-
PC2	50	-	TDI	-	-	-	-	-	T5CCP0	-	-	-	-
PC3	49	-	TDO SWO	-	-	-	-	-	T5CCP1	-	-	-	-
PC4	16	C1-	U4Rx	U1Rx	-	-	-	-	WT0CCP0	U1RTS	-	-	-
PC5	15	C1+	U4Tx	U1Tx	-	-	-	-	WT0CCP1	U1CTS	-	-	-
PC6	14	C0+	U3Rx	-	-	-	-	-	WT1CCP0	-	-	-	-
PC7	13	C0-	U3Tx	-	-	-	-	-	WT1CCP1	-	-	-	-
PD0	61	AIN7	SSI3Clk	SSI1Clk	I2C3SCL	-	-	-	WT2CCP0	-	-	-	-
PD1	62	AIN6	SSI3Fss	SSI1Fss	I2C3SDA	-	-	-	WT2CCP1	-	-	-	-
PD2	63	AIN5	SSI3Rx	SSI1Rx	-	-	-	-	WT3CCP0	-	-	-	-
PD3	64	AIN4	SSI3Tx	SSI1Tx	-	-	-	-	WT3CCP1	-	-	-	-
PD4	43	USB0DM	U6Rx	-	-	-	-	-	WT4CCP0	-	-	-	-
PD5	44	USB0DP	U6Tx	-	-	-	-	-	WT4CCP1	-	-	-	-
PD6	53	-	U2Rx	-	-	-	-	-	WT5CCP0	-	-	-	-
PD7	10	-	U2Tx	-	-	-	-	-	WT5CCP1	NMI	-	-	-
PE0	9	AIN3	U7Rx	-	-	-	-	-	-	-	-	-	-
PE1	8	AIN2	U7Tx	-	-	-	-	-	-	-	-	-	-
PE2	7	AIN1	-	-	-	-	-	-	-	-	-	-	-
PE3	6	AIN0	-	-	-	-	-	-	-	-	-	-	-
PE4	59	AIN9	U5Rx	-	I2C2SCL	-	-	-	-	CAN0Rx	-	-	-
PE5	60	AIN8	U5Tx	-	I2C2SDA	-	-	-	-	CAN0Tx	-	-	-
PF0	28	-	U1RTS	SSI1Rx	CAN0Rx	-	-	-	T0CCP0	NMI	C0o	-	-
PF1	29	-	U1CTS	SSI1Tx	-	-	-	-	T0CCP1	-	C1o	TRD1	-
PF2	30	-	-	SSI1Clk	-	-	-	-	T1CCP0	-	-	TRD0	-

Table 20-5. GPIO Pins and Alternate Functions *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]									
			1	2	3	4	5	6	7	8	9	14	15
PF3	31	-	-	SSI1Fss	CAN0Tx	-	-	-	T1CCP1	-	-	TRCLK	-
PF4	5	-	-	-	-	-	-	-	T2CCP0	-	-	-	-
PG0	37	-	-	-	I2C3SCL	-	-	-	T4CCP0	-	-	-	-
PG1	36	-	-	-	I2C3SDA	-	-	-	T4CCP1	-	-	-	-
PG2	35	-	-	-	I2C4SCL	-	-	-	T5CCP0	-	-	-	-
PG3	34	-	-	-	I2C4SDA	-	-	-	T5CCP1	-	-	-	-
PG4	33	-	U2Rx	-	I2C1SCL	-	-	-	WT0CCP0	-	-	-	-
PG5	32	-	U2Tx	-	I2C1SDA	-	-	-	WT0CCP1	-	-	-	-

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin. Encodings 10-13 are not used on this device.

20.5 Possible Pin Assignments for Alternate Functions

Table 20-6. Possible Pin Assignments for Alternate Functions

# of Possible Assignments	Alternate Function	GPIO Function
one	AIN0	PE3
	AIN1	PE2
	AIN10	PB4
	AIN11	PB5
	AIN2	PE1
	AIN3	PE0
	AIN4	PD3
	AIN5	PD2
	AIN6	PD1
	AIN7	PD0
	AIN8	PE5
	AIN9	PE4
	C0+	PC6
	C0-	PC7
	C0o	PF0
	C1+	PC5
	C1-	PC4
	C1o	PF1
	I2C0SCL	PB2
	I2C0SDA	PB3
	I2C2SCL	PE4
	I2C2SDA	PE5
	I2C4SCL	PG2
	I2C4SDA	PG3
	I2C5SCL	PB6
	I2C5SDA	PB7
	SSI0Clk	PA2
	SSI0Fss	PA3
	SSI0Rx	PA4
	SSI0Tx	PA5
	SSI2Clk	PB4
	SSI2Fss	PB5
	SSI2Rx	PB6
	SSI2Tx	PB7
	SSI3Clk	PD0
	SSI3Fss	PD1
	SSI3Rx	PD2
	SSI3Tx	PD3
	SWCLK	PC0

Table 20-6. Possible Pin Assignments for Alternate Functions *(continued)*

# of Possible Assignments	Alternate Function	GPIO Function
	SWDIO	PC1
	SWO	PC3
	T2CCP1	PB1
	T3CCP0	PB2
	T3CCP1	PB3
	TCK	PC0
	TDI	PC2
	TDO	PC3
	TMS	PC1
	TRCLK	PF3
	TRD0	PF2
	TRD1	PF1
	U0Rx	PA0
	U0Tx	PA1
	U3Rx	PC6
	U3Tx	PC7
	U4Rx	PC4
	U4Tx	PC5
	U5Rx	PE4
	U5Tx	PE5
	U6Rx	PD4
	U6Tx	PD5
	U7Rx	PE0
	U7Tx	PE1
	USB0DM	PD4
	USB0DP	PD5
	WT1CCP0	PC6
	WT1CCP1	PC7
	WT2CCP0	PD0
	WT2CCP1	PD1
	WT3CCP0	PD2
	WT3CCP1	PD3
	WT4CCP0	PD4
	WT4CCP1	PD5
	WT5CCP0	PD6
	WT5CCP1	PD7

Table 20-6. Possible Pin Assignments for Alternate Functions *(continued)*

# of Possible Assignments	Alternate Function	GPIO Function
two	I2C1SCL	PA6 PG4
	I2C1SDA	PA7 PG5
	I2C3SCL	PD0 PG0
	I2C3SDA	PD1 PG1
	NMI	PD7 PF0
	SSI1Clk	PD0 PF2
	SSI1Fss	PD1 PF3
	SSI1Rx	PD2 PF0
	SSI1Tx	PD3 PF1
	T0CCP0	PB6 PF0
	T0CCP1	PB7 PF1
	T1CCP0	PB4 PF2
	T1CCP1	PB5 PF3
	T2CCP0	PB0 PF4
	T4CCP0	PC0 PG0
	T4CCP1	PC1 PG1
	T5CCP0	PC2 PG2
	T5CCP1	PC3 PG3
	U1CTS	PC5 PF1
	U1RTS	PC4 PF0
	U1Rx	PB0 PC4
	U1Tx	PB1 PC5
	U2Rx	PD6 PG4
	U2Tx	PD7 PG5
	WT0CCP0	PC4 PG4
	WT0CCP1	PC5 PG5
three	CAN0Rx	PB4 PE4 PF0
	CAN0Tx	PB5 PE5 PF3

20.6 Connections for Unused Signals

Table 20-7 on page 1120 shows how to handle signals for functions that are not used in a particular system implementation for devices that are in a 64-pin LQFP package. Two options are shown in the table: an acceptable practice and a preferred practice for reduced power consumption and improved EMC characteristics. If a module is not used in a system, and its inputs are grounded, it is important that the clock to the module is never enabled by setting the corresponding bit in the **RCGCx** register.

Table 20-7. Connections for Unused Signals (64-Pin LQFP)

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
GPIO	All unused GPIOs	-	NC	GND
No Connects	NC	See NC pin numbers in Table 20-3 on page 1105	NC	NC

Table 20-7. Connections for Unused Signals (64-Pin LQFP) *(continued)*

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
System Control	OSC0	40	NC	GND
	OSC1	41	NC	NC
	$\overline{\texttt{RST}}$	38	VDD	Pull up as shown in Figure 5-1 on page 204
USB	USB0DM	43	NC	GND
	USB0DP	44	NC	GND

21 Electrical Characteristics

21.1 Maximum Ratings

The maximum ratings are the limits to which the device can be subjected without permanently damaging the device. Device reliability may be adversely affected by exposure to absolute-maximum ratings for extended periods.

Note: The device is not guaranteed to operate properly at the maximum ratings.

Table 21-1. Absolute Maximum Ratings

Parameter	Parameter Name[a]	Value		Unit
		Min	Max	
V_{DD}	V_{DD} supply voltage	0	4	V
V_{DDA}	V_{DDA} supply voltage[b]	0	4	V
V_{BATRMP}	V_{BAT} battery supply voltage ramp time	0	0.7	V/µs
V_{IN_GPIO}	Input voltage on GPIOs, regardless of whether the microcontroller is powered[cde]	-0.3	5.5	V
	Input voltage for PD4, PD5, PB0 and PB1 when configured as GPIO	-0.3	V_{DD} + 0.3	V
$I_{GPIOMAX}$	Maximum current per output pin	-	25	mA
T_S	Unpowered storage temperature range	-65	150	°C
T_{JMAX}	Maximum junction temperature	-	150	°C

a. Voltages are measured with respect to GND.

b. To ensure proper operation, VDDA must be powered before VDD if sourced from different supplies, or connected to the same supply as VDD. Note that the minimum operating voltage for VDD differs from the minimum operating voltage for VDDA. This change should be accounted for in the system design if both are sourced from the same supply. There is not a restriction on order for powering off.

c. Applies to static and dynamic signals including overshoot.

d. Refer to Figure 21-15 on page 1148 for a representation of the ESD protection on GPIOs.

e. For additional details, see the note on GPIO pad tolerance in “GPIO Module Characteristics” on page 1147.

Important: This device contains circuitry to protect the I/Os against damage due to high-static voltages; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (see “Connections for Unused Signals” on page 1120).

Table 21-2. ESD Absolute Maximum Ratings

	Parameter	Min	Nom	Max	Unit
Component-Level ESD Stress Voltage[a]	V_{ESDHBM} [b]	-	-	2.0	kV
	V_{ESDCDM} [c]	-	-	500	V

a. Electrostatic discharge (ESD) to measure device sensitivity/immunity to damage caused by electrostatic discharges in device.

b. Level listed is passing level per ANSI/ESDA/JEDEC JS-001. JEDEC document JEP155 states that 500V HBM allows safe manufacturing with a standard ESD control process.

c. Level listed is the passing level per EIA-JEDEC JESD22-C101E. JEDEC document JEP157 states that 250V CDM allows safe manufacturing with a standard ESD control process.

21.2 Operating Characteristics

Table 21-3. Temperature Characteristics

Characteristic	Symbol	Value	Unit
Ambient operating temperature range	T_A	-40 to +85	°C
Case operating temperature range	T_C	-40 to +93	°C
Junction operating temperature range	T_J	-40 to +96	°C

Table 21-4. Thermal Characteristics[a]

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient)[b]	Θ_{JA}	54.8	°C/W
Thermal resistance (junction to board)[b]	Θ_{JB}	27.5	°C/W
Thermal resistance (junction to case)[b]	Θ_{JC}	15.8	°C/W
Thermal metric (junction to top of package)	Ψ_{JT}	0.7	°C/W
Thermal metric (junction to board)	Ψ_{JB}	27.1	°C/W
Junction temperature formula	T_J	$T_C + (P \bullet \Psi_{JT})$ $T_{PCB} + (P \bullet \Psi_{JB})$[c] $T_A + (P \bullet \Theta_{JA})$[d] $T_B + (P \bullet \Theta_{JB})$[ef]	°C

[a] For more details about thermal metrics and definitions, see the *Semiconductor and IC Package Thermal Metrics Application Report* (literature number SPRA953).

[b] Junction to ambient thermal resistance (Θ_{JA}), junction to board thermal resistance (Θ_{JB}), and junction to case thermal resistance (Θ_{JC}) numbers are determined by a package simulator.

[c] T_{PCB} is the temperature of the board acquired by following the steps listed in the EAI/JESD 51-8 standard summarized in the *Semiconductor and IC Package Thermal Metrics Application Report* (literature number SPRA953).

[d] Because Θ_{JA} is highly variable and based on factors such as board design, chip/pad size, altitude, and external ambient temperature, it is recommended that equations containing Ψ_{JT} and Ψ_{JB} be used for best results.

[e] T_B is temperature of the board.

[f] Θ_{JB} is not a pure reflection of the internal resistance of the package because it includes the resistance of the testing board and environment. It is recommended that equations containing Ψ_{JT} and Ψ_{JB} be used for best results.

21.3 Recommended Operating Conditions

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the V_{OL} value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package with the total number of high-current GPIO outputs not exceeding four for the entire package.

Table 21-5. Recommended DC Operating Conditions

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{DD}	V_{DD} supply voltage	3.15	3.3	3.63	V
V_{DDA}	V_{DDA} supply voltage	2.97	3.3	3.63	V
V_{DDC}	V_{DDC} supply voltage	1.08	1.2	1.32	V
V_{DDCDS}[ab]	V_{DDC} supply voltage, Deep-sleep mode	1.08	-	1.32	V

[a] These values are valid when LDO is in operation.

[b] There are peripheral timing restrictions for SSI and LPC in Deep-sleep mode. Please refer to those peripheral characteristic sections for more information.

Table 21-6. Recommended GPIO Pad Operating Conditions

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{IH}	GPIO high-level input voltage	0.65 * V_{DD}	-	5.5	V
V_{IL}	GPIO low-level input voltage	0	-	0.35 * V_{DD}	V
V_{HYS}	GPIO input hysteresis	0.2	-	-	V
V_{OH}	GPIO high-level output voltage	2.4	-	-	V
V_{OL}	GPIO low-level output voltage	-	-	0.4	V
I_{OH}	High-level source current, V_{OH}=2.4 V[a]				
	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[a]				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA
	8-mA Drive, V_{OL}=1.2 V	18.0	-	-	mA

[a] I_O specifications reflect the maximum current where the corresponding output voltage meets the V_{OH}/V_{OL} thresholds. I_O current can exceed these limits (subject to absolute maximum ratings).

Table 21-7. GPIO Current Restrictions[a]

Parameter	Parameter Name	Min	Nom	Max	Unit
I_{MAXL}	Cumulative maximum GPIO current per side, left[b]	-	-	30	mA
I_{MAXB}	Cumulative maximum GPIO current per side, bottom[b]	-	-	35	mA
I_{MAXR}	Cumulative maximum GPIO current per side, right[b]	-	-	40	mA
I_{MAXT}	Cumulative maximum GPIO current per side, top[b]	-	-	40	mA

[a] Based on design simulations, not tested in production.

[b] Sum of sink and source current for GPIOs as shown in Table 21-8 on page 1125.

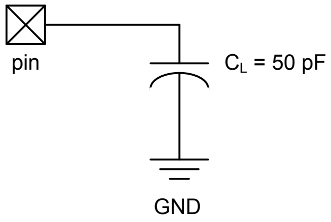
Table 21-8. GPIO Package Side Assignments

Side	GPIOs
Left	PB[6-7], PC[4-7], PD7, PE[0-3], PF4
Bottom	PA[0-7], PF[0-3], PG5
Right	PB[0-3], PD[4-5], PG[0-4]
Top	PB[4-5], PC[0-3], PD[0-3,6], PE[4-5]

21.4 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements.

Figure 21-1. Load Conditions



21.5 JTAG and Boundary Scan

Table 21-9. JTAG Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	F_{TCK}	TCK operational clock frequency[a]	0	-	10	MHz
J2	T_{TCK}	TCK operational clock period	100	-	-	ns
J3	T_{TCK_LOW}	TCK clock Low time	-	$t_{TCK}/2$	-	ns
J4	T_{TCK_HIGH}	TCK clock High time	-	$t_{TCK}/2$	-	ns
J5	T_{TCK_R}	TCK rise time	0	-	10	ns
J6	T_{TCK_F}	TCK fall time	0	-	10	ns
J7	T_{TMS_SU}	TMS setup time to TCK rise	8	-	-	ns
J8	T_{TMS_HLD}	TMS hold time from TCK rise	4	-	-	ns
J9	T_{TDI_SU}	TDI setup time to TCK rise	18	-	-	ns
J10	T_{TDI_HLD}	TDI hold time from TCK rise	4	-	-	ns
J11	T_{TDO_ZDV}	TCK fall to Data Valid from High-Z, 2-mA drive	-	13	35	ns
		TCK fall to Data Valid from High-Z, 4-mA drive		9	26	ns
		TCK fall to Data Valid from High-Z, 8-mA drive		8	26	ns
		TCK fall to Data Valid from High-Z, 8-mA drive with slew rate control		10	29	ns
J12	T_{TDO_DV}	TCK fall to Data Valid from Data Valid, 2-mA drive	-	14	20	ns
		TCK fall to Data Valid from Data Valid, 4-mA drive		10	26	ns
		TCK fall to Data Valid from Data Valid, 8-mA drive		8	21	ns
		TCK fall to Data Valid from Data Valid, 8-mA drive with slew rate control		10	26	ns
J13	T_{TDO_DVZ}	TCK fall to High-Z from Data Valid, 2-mA drive	-	7	16	ns
		TCK fall to High-Z from Data Valid, 4-mA drive		7	16	ns
		TCK fall to High-Z from Data Valid, 8-mA drive		7	16	ns
		TCK fall to High-Z from Data Valid, 8-mA drive with slew rate control		8	19	ns

[a] A ratio of at least 8:1 must be kept between the system clock and TCK.

Figure 21-2. JTAG Test Clock Input Timing

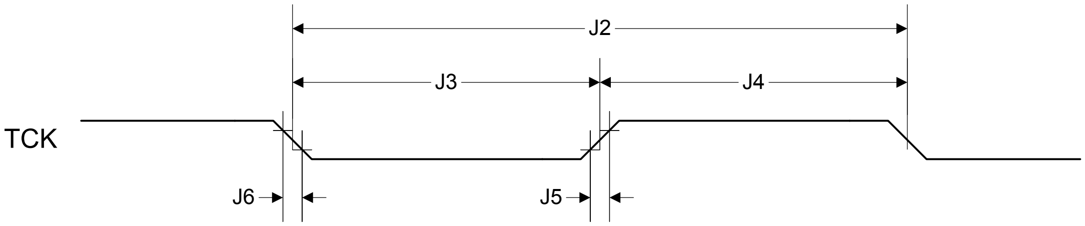
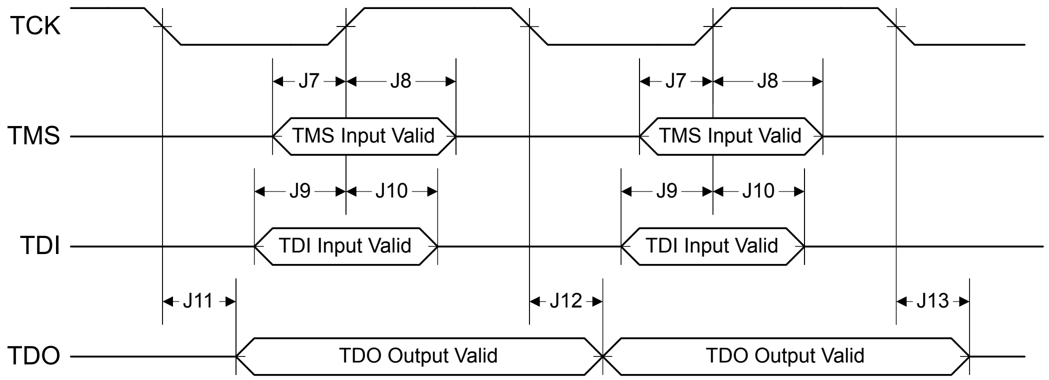


Figure 21-3. JTAG Test Access Port (TAP) Timing



21.6 Power and Brown-Out

Table 21-10. Power-On and Brown-Out Levels

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
P1	T_{VDDA_RISE}	Analog Supply Voltage (VDDA) Rise Time	-	-	∞	µs
P2	T_{VDD_RISE}	I/O Supply Voltage (VDD) Rise Time	-	-	∞	µs
P3	T_{VDDC_RISE}[a]	Core Supply Voltage (VDDC) Rise Time	10.00	-	150.00	µs
P4	V_{POR}	Power-On Reset Threshold	2.00	2.30	2.60	V
P5	V_{VDDA_POK}	VDDA Power-OK Threshold (Rising Edge)	2.70	2.85	3.00	V
		VDDA Power-OK Threshold (Falling Edge)	2.71	2.80	2.89	V
P6	V_{VDD_POK}[b]	VDD Power-OK Threshold (Rising Edge)	2.85	3.00	3.15	V
		VDD Power-OK Threshold (Falling Edge)	2.70	2.78	2.87	V
P7	V_{VDD_BOR0}	Brown-Out 0 Reset Threshold	2.93	3.02	3.11	V
P8	V_{VDD_BOR1}	Brown-Out 1 Reset Threshold	2.83	2.92	3.01	V
P9	V_{VDDC_POK}	VDDC Power-OK Threshold (Rising Edge)	0.80	0.95	1.10	V
		VDDC Power-OK Threshold (Falling Edge)	0.71	0.80	0.89	V

[a] The MIN and MAX values are guaranteed by design assuming the external filter capacitor load is within the range of CLDO. Please refer to “On-Chip Low Drop-Out (LDO) Regulator” on page 1137 for the CLDO value.

[b] Digital logic, Flash memory, and SRAM are all designed to operate at VDD voltages below 2.70 V. The internal POK reset protects the device from unpredictable operation on power down.

21.6.1 VDDA Levels

The V_{DDA} supply has two monitors:

- Power-On Reset (POR)
- Power-OK (POK)

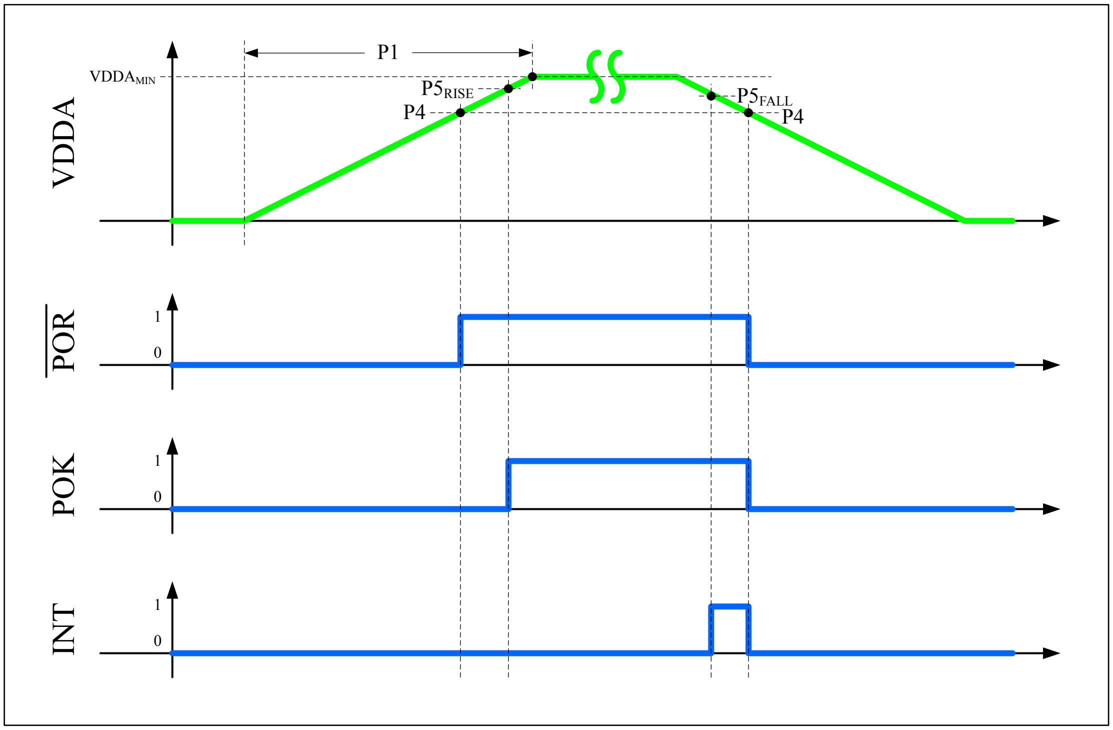
The POR monitor is used to keep the analog circuitry in reset until the V_{DDA} supply has reached the correct range for the analog circuitry to begin operating. The POK monitor is used to keep the digital circuitry in reset until the V_{DDA} power supply is at an acceptable operational level. The digital Power-On Reset (Digital POR) is only released when the Power-On Reset has deasserted and all of the Power-OK monitors for each of the supplies indicate that power levels are in operational ranges.

Once the V_{DDA} POK monitor has released the digital Power-On Reset on the initial power-up, voltage drops on the V_{DDA} supply will only be reflected in the following bits. The digital Power-On Reset will not be re-asserted.

- VDDARIS bit in the **Raw Interrupt Status (RIS)** register (see page 231).
- VDDAMIS bit in the **Masked Interrupt Status and Clear (MISC)** register (see page 236). This bit is set only if the VDDAIM bit in the **Interrupt Mask Control (IMC)** register has been set.

Figure 21-4 on page 1130 shows the relationship between V_{DDA}, POR, POK, and an interrupt event.

Figure 21-4. Power Assertions versus VDDA Levels



21.6.2 VDD Levels

The V_{DD} supply has three monitors:

- Power-OK (POK)
- Brown-Out Reset0 (BOR0)
- Brown-Out Reset1 (BOR1)

The POK monitor is used to keep the digital circuitry in reset until the V_{DD} power supply is at an acceptable operational level. The digital Power-On Reset ($\overline{\texttt{Digital POR}}$) is only released when the Power-On Reset has deasserted and all of the Power-OK monitors for each of the supplies indicate that power levels are in operational ranges. The BOR0 and the BOR1 monitors are used to generate a reset to the device or assert an interrupt if the V_{DD} supply drops below its operational range. The BOR1 monitor's threshold is in between the BOR0 and POK thresholds.

If either a BOR0 event or a BOR1 event occurs, the following bits are affected:

- `BOR0RIS` or `BOR1RIS` bits in the **Raw Interrupt Status (RIS)** register (see page 231).
- `BOR0MIS` or `BOR1MIS` bits in the **Masked Interrupt Status and Clear (MISC)** register (see page 236). These bits are set only if the respective `BOR0IM` or `BOR1IM` bits in the **Interrupt Mask Control (IMC)** register have been set.
- `BOR` bit in the **Reset Cause (RESC)** register (see page 239). This bit is set only if either of the BOR0 or BOR1 events have been configured to initiate a reset.

In addition, the following bits control both the BOR0 and BOR1 events:

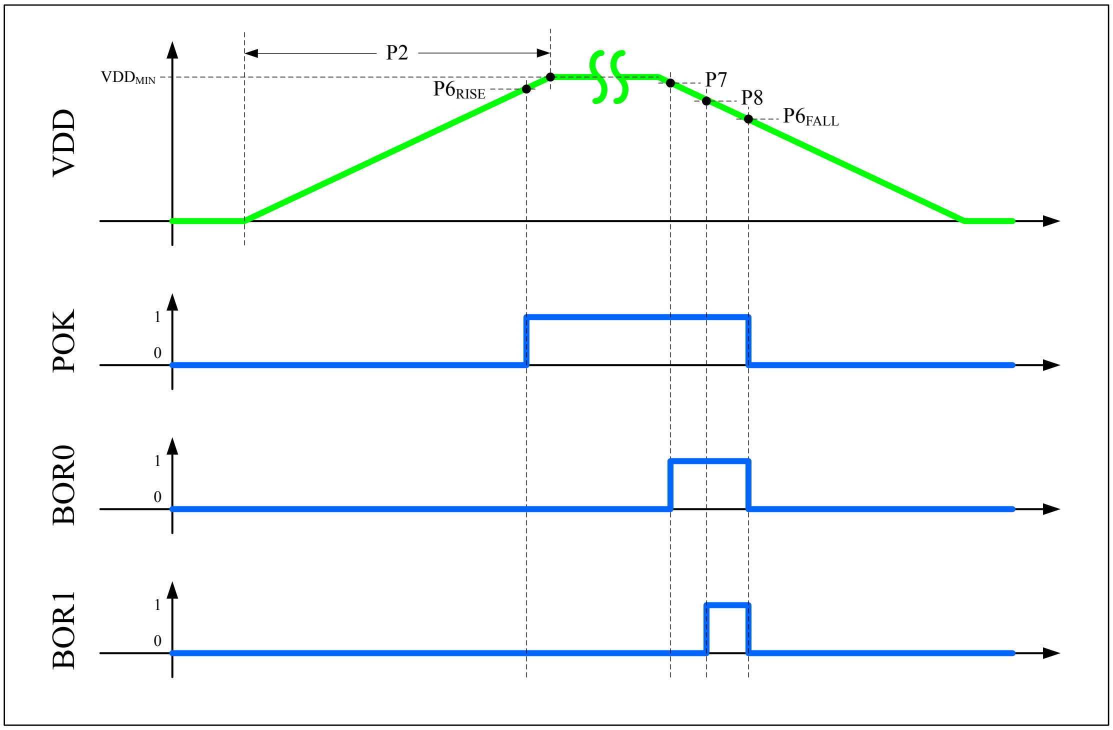
- `BOR0IM` or `BOR1IM` bits in the **Interrupt Mask Control (IMC)** register (see page 234).

- BOR0 or BOR1 bits in the **Power-On and Brown-Out Reset Control (PBORCTL)** register (see page 230).

Figure 21-5 on page 1131 shows the relationship between:

- V_{DD}, POK, and a BOR0 event
- V_{DD}, POK, and a BOR1 event

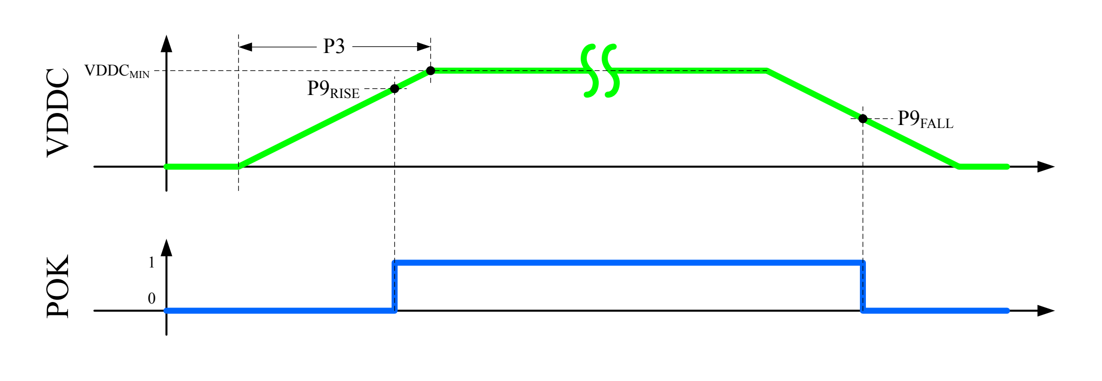
Figure 21-5. Power and Brown-Out Assertions versus VDD Levels



21.6.3 VDDC Levels

The V_{DDC} supply has one monitor: the Power-OK (POK). The POK monitor is used to keep the digital circuitry in reset until the V_{DDC} power supply is at an acceptable operational level. The digital Power-On Reset ($\overline{\text{Digital POR}}$) is only released when the Power-On Reset has deasserted and all of the Power-OK monitors for each of the supplies indicate that power levels are in operational ranges. Figure 21-6 on page 1132 shows the relationship between POK and V_{DDC}.

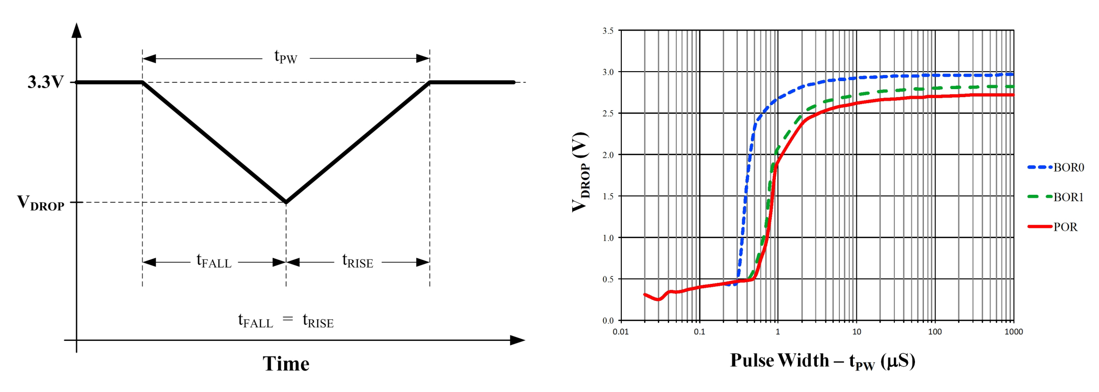
Figure 21-6. POK assertion vs VDDC



21.6.4 VDD Glitches

Figure 21-7 on page 1132 shows the response of the BOR0, BOR1, and the POR circuit to glitches on the V_{DD} supply.

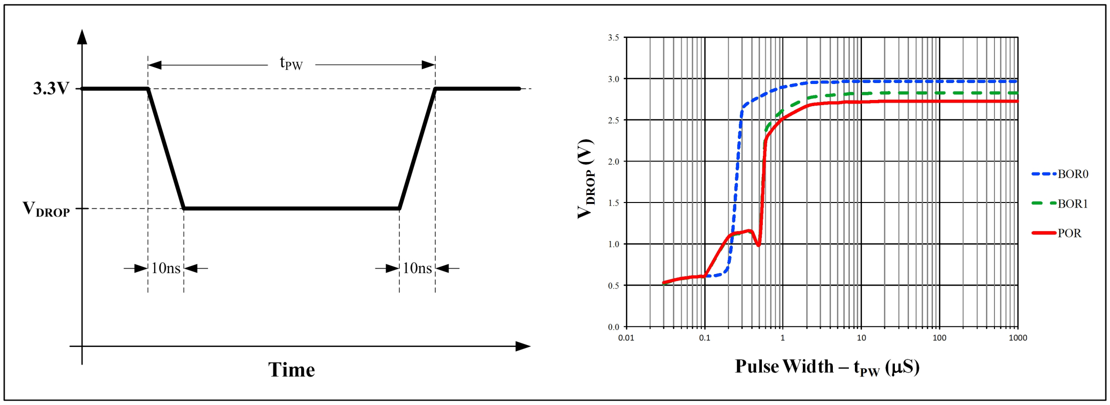
Figure 21-7. POR-BOR0-BOR1 VDD Glitch Response



21.6.5 VDD Droop Response

Figure 21-8 on page 1133 shows the response of the BOR0, BOR1, and the POR monitors to a drop on the V_{DD} supply.

Figure 21-8. POR-BOR0-BOR1 VDD Droop Response



21.7 Reset

Table 21-11. Reset Characteristics

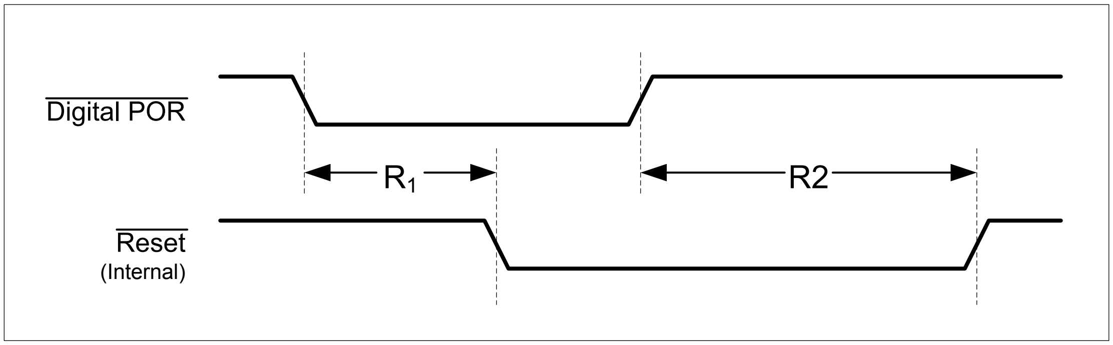
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	$T_{DPORDLY}$	Digital POR to Internal Reset assertion delay[a]	0.80	-	5.35	µs
R2	T_{IRTOUT}	Standard Internal Reset time	-	9	11.5	ms
		Internal Reset time with recovery code repair (program or erase)[b]	-	-	6400[c]	ms
R3	$T_{BOR0DLY}$	BOR0 to Internal Reset assertion delay[a]	0.25	-	1.95	µs
R3	$T_{BOR1DLY}$	BOR1 to Internal Reset assertion delay[a]	0.75	-	5.95	µs
R4	T_{RSTMIN}	Minimum $\overline{RST}$ pulse width	-	250	-	ns
R5	$T_{IRHWDLY}$	$\overline{RST}$ to Internal Reset assertion delay	-	250	-	ns
R6	T_{IRSWR}	Internal reset timeout after software-initiated system reset	-	2.07	-	µs
R7	T_{IRWDR}	Internal reset timeout after Watchdog reset	-	2.10	-	µs
R8	T_{IRMFR}	Internal reset timeout after MOSC failure reset	-	1.92	-	µs

a. Timing values are dependent on the V_{DD} power-down ramp rate.

b. This parameter applies only in situations where a power-loss or brown-out event occurs during an EEPROM program or erase operation, and EEPROM needs to be repaired (which is a rare case). For all other sequences, there is no impact to normal Power-On Reset (POR) timing. This delay is in addition to other POR delays.

c. This value represents the maximum internal reset time when the EEPROM reaches its endurance limit.

Figure 21-9. Digital Power-On Reset Timing



Note: The digital Power-On Reset is only released when the analog Power-On Reset has deasserted and all of the Power-OK monitors for each of the supplies indicate that power levels are in operational ranges.

Figure 21-10. Brown-Out Reset Timing

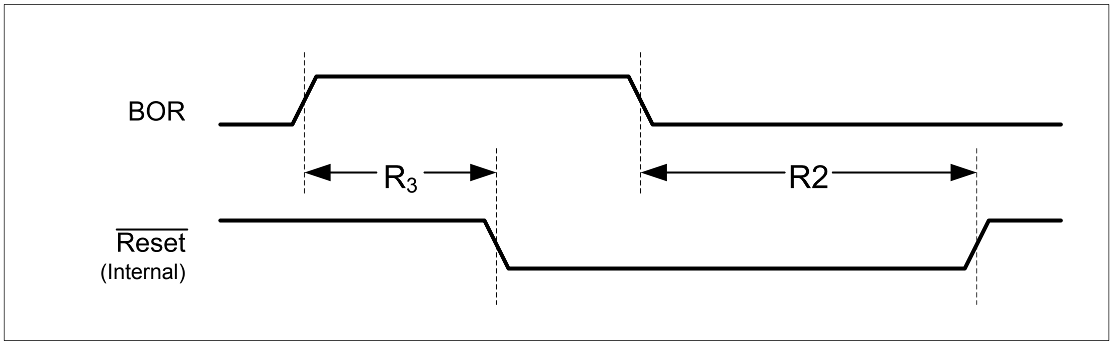


Figure 21-11. External Reset Timing ($\overline{\text{RST}}$)

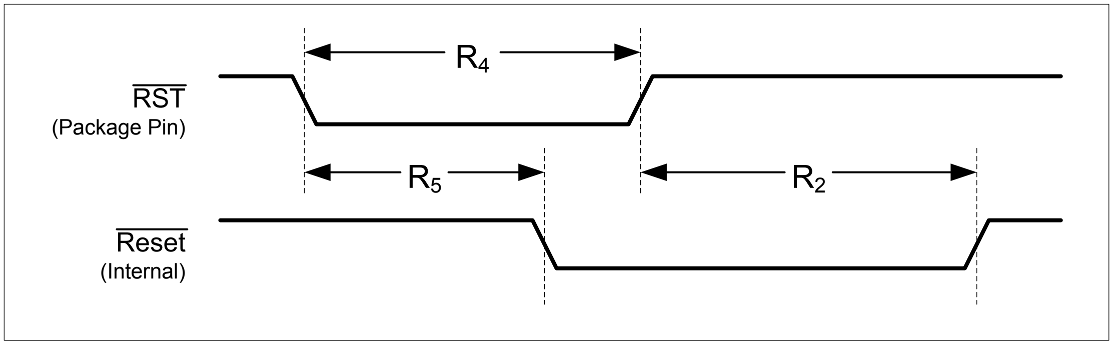


Figure 21-12. Software Reset Timing

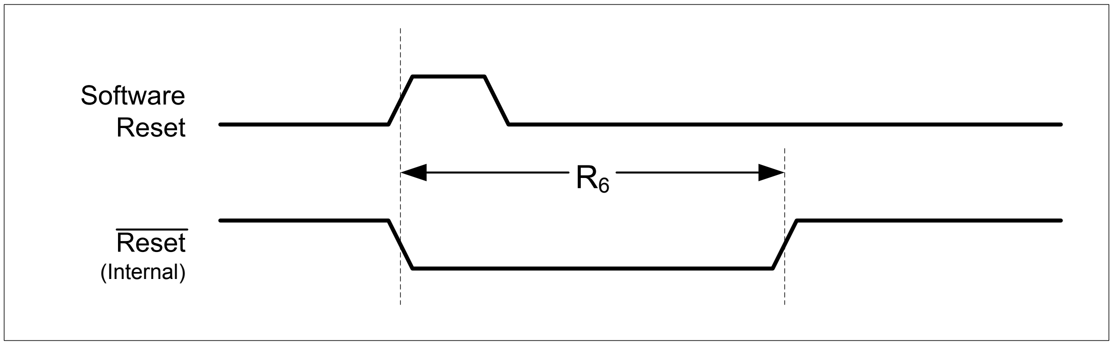


Figure 21-13. Watchdog Reset Timing

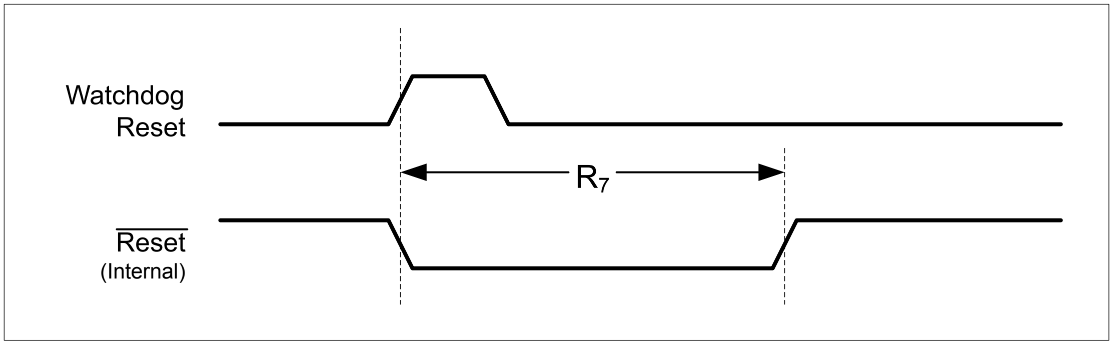
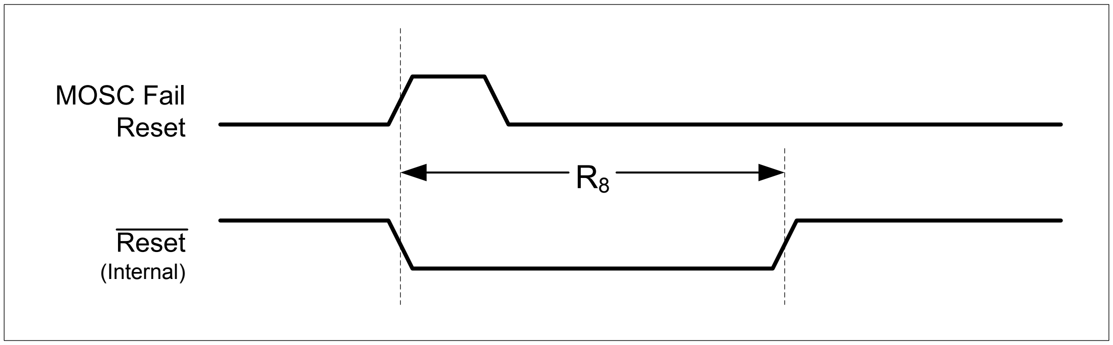


Figure 21-14. MOSC Failure Reset Timing



21.8 On-Chip Low Drop-Out (LDO) Regulator

Table 21-12. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
C_{LDO}	External filter capacitor size for internal power supply[a]	2.5	-	4.0	μF
ESR	Filter capacitor equivalent series resistance	10	-	100	mΩ
ESL	Filter capacitor equivalent series inductance	-	-	0.5	nH
V_{LDO}	LDO output voltage	1.08	1.2	1.32	V
I_{INRUSH}	Inrush current	50	-	250	mA

[a] The capacitor should be connected as close as possible to pin 56.

21.9 Clocks

The following sections provide specifications on the various clock sources and mode.

21.9.1 PLL Specifications

The following tables provide specifications for using the PLL.

Table 21-13. Phase Locked Loop (PLL) Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{REF_XTAL}	Crystal reference	5[a]	-	25	MHz
F_{REF_EXT}	External clock reference[a]	5[a]	-	25	MHz
F_{PLL}	PLL frequency[b]	-	400	-	MHz
T_{READY}	PLL lock time, enabling the PLL	-	-	512 * (N+1)[c]	reference clocks[d]
	PLL lock time, changing the XTAL field in the **RCC**/**RCC2** register or changing the OSCSRC between MOSC and PIOSC	-	-	128 * (N+1)[c]	reference clocks[d]

a. If the PLL is not used, the minimum input frequency can be 4 MHz.

b. PLL frequency is automatically calculated by the hardware based on the XTAL field of the **RCC** register. The PLL frequency that is set by the hardware can be calculated using the values in the **PLLFREQ0** and **PLLFREQ1** registers.

c. N is the value in the N field in the **PLLFREQ1** register.

d. A reference clock is the clock period of the crystal being used, which can be MOSC or PIOSC. For example, a 16-MHz crystal connected to MOSC yields a reference clock of 62.5 ns.

Table 21-14 on page 1138 shows the actual frequency of the PLL based on the crystal frequency used (defined by the XTAL field in the **RCC** register).

Table 21-14. Actual PLL Frequency

XTAL	Crystal Frequency (MHz)	MINT	MFRAC	Q	N	PLL Multiplier	PLL Frequency (MHz)	Error
0x09	5.0	0x50	0x0	0x0	0x0	80	400	-
0x0A	5.12	0x9C	0x100	0x0	0x1	156.25	400	-
0x0B	6.0	0xC8	0x0	0x0	0x2	200	400	-
0x0C	6.144	0xC3	0x140	0x0	0x2	195.3125	400	-
0x0D	7.3728	0xA2	0x30A	0x0	0x2	162.7598	399.9984	0.0004%
0x0E	8.0	0x32	0x0	0x0	0x0	50	400	-
0x0F	8.192	0xC3	0x140	0x0	0x3	195.3125	400	-
0x10	10.0	0x50	0x0	0x0	0x1	80	400	-
0x11	12.0	0xC8	0x0	0x0	0x5	200	400	-
0x12	12.288	0xC3	0x140	0x0	0x5	195.3125	400	-
0x13	13.56	0xB0	0x3F6	0x0	0x5	176.9902	399.9979	0.0005%
0x14	14.318	0xC3	0x238	0x0	0x6	195.5547	399.9982	0.0005%
0x15	16.0	0x32	0x0	0x0	0x1	50	400	-
0x16	16.384	0xC3	0x140	0x0	0x7	195.3125	400	-
0x17	18	0xC8	0x0	0x0	0x8	200	400	-
0x18	20	0x50	0x0	0x0	0x3	80	400	-
0x19	24	0x32	0x0	0x0	0x2	50	400	-

Table 21-14. Actual PLL Frequency *(continued)*

XTAL	Crystal Frequency (MHz)	MINT	MFRAC	Q	N	PLL Multiplier	PLL Frequency (MHz)	Error
0x1A	25	0x50	0x0	0x0	0x4	80	400	-

21.9.2 PIOSC Specifications

Table 21-15. PIOSC Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{PIOSC}	Factory calibration: Internal 16-MHz precision oscillator frequency variance across the specified voltage and temperature range when factory calibration is used	-	-	±3%	-
	Recalibration: Internal 16-MHz precision oscillator frequency variance when 7-bit recalibration is used	-	-	±1%[a]	-
T_{START}	PIOSC startup time[b]	-	-	1	µs

a. ±1% is only guaranteed at the specific voltage/temperature condition where the recalibration occurs.

b. PIOSC startup time is part of reset and is included in the internal reset timeout value (T_{IRTOUT}) given in Table 21-11 on page 1134. Note that the T_{START} value is based on simulation.

21.9.3 Low-Frequency Internal Oscillator (LFIOSC) Specifications

Table 21-16. Low-Frequency internal Oscillator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{LFIOSC}	Low-frequency internal oscillator (LFIOSC) frequency	10	33	90	KHz

21.9.4 Main Oscillator Specifications

Table 21-17. Main Oscillator Input Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{MOSC}	Parallel resonance frequency	4[a]	-	25	MHz
C_1, C_2	External load capacitance on OSC0, OSC1 pins[b]	10	-	24	pF
C_{PKG}	Device package stray shunt capacitance[b]	-	0.5	-	pF
C_{PCB}	PCB stray shunt capacitance[b]	-	0.5	-	pF
C_{SHUNT}	Total shunt capacitance[b]	-	-	4	pF
ESR	Crystal effective series resistance, 4 MHz[cd]	-	-	300	Ω
	Crystal effective series resistance, 6 MHz[cd]	-	-	200	Ω
	Crystal effective series resistance, 8 MHz[cd]	-	-	130	Ω
	Crystal effective series resistance, 12 MHz[cd]	-	-	120	Ω
	Crystal effective series resistance, 16 MHz[cd]	-	-	100	Ω
	Crystal effective series resistance, 25 MHz[cd]	-	-	50	Ω
DL	Oscillator output drive level[e]	-	OSC_{PWR}	-	mW
T_{START}	Oscillator startup time, when using a crystal[f]	-	-	18	ms

Table 21-17. Main Oscillator Input Characteristics *(continued)*

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{IH}	CMOS input high level, when using an external oscillator	0.65 * V_{DD}	-	V_{DD}	V
V_{IL}	CMOS input low level, when using an external oscillator	GND	-	0.35 * V_{DD}	V
V_{HYS}	CMOS input buffer hysteresis, when using an external oscillator	150	-	-	mV
DC_{OSC_EXT}	External clock reference duty cycle	45	-	55	%

a. 5 MHz is the minimum when using the PLL.

b. See information below table.

c. Crystal ESR specified by crystal manufacturer.

d. Crystal vendors can be contacted to confirm these specifications are met for a specific crystal part number if the vendors generic crystal datasheet show limits outside of these specifications.

e. OSC_{PWR} = (2 * pi * F_P * C_L * 2.5)^2 * ESR / 2. An estimation of the typical power delivered to the crystal is based on the C_L, F_P and ESR parameters of the crystal in the circuit as calculated by the OSC_{PWR} equation. Ensure that the value calculated for OSC_{PWR} does not exceed the crystal's drive-level maximum.

f. Oscillator startup time is specified from the time the oscillator is enabled to when it reaches a stable point of oscillation such that the internal clock is valid.

The load capacitors added on the board, C_1 and C_2, should be chosen such that the following equation is satisfied (see Table 21-17 on page 1139 for typical values and Table 21-18 on page 1141 for detailed crystal parameter information).

- C_L = load capacitance specified by crystal manufacturer
- $C_L = (C_1*C_2)/(C_1+C_2) + C_{SHUNT}$
- $C_{SHUNT} = C_0 + C_{PKG} + C_{PCB}$ (total shunt capacitance seen across OSC0, OSC1 crystal inputs)
- C_{PKG}, C_{PCB} = the mutual caps as measured across the OSC0,OSC1 pins excluding the crystal.
- C_0 = Shunt capacitance of crystal specified by the crystal manufacturer

Table 21-18 on page 1141 lists part numbers of crystals that have been simulated and confirmed to operate within the specifications in Table 21-17 on page 1139. Other crystals that have nearly identical crystal parameters can be expected to work as well.

In the table below, the crystal parameters labeled C0, C1 and L1 are values that are obtained from the crystal manufacturer. These numbers are usually a result of testing a relevant batch of crystals on a network analyzer. The parameters labeled ESR, DL and C_L are maximum numbers usually available in the data sheet for a crystal.

The table also includes three columns of Recommended Component Values. These values apply to system board components. C_1 and C_2 are the values in pico Farads of the load capacitors that should be put on each leg of the crystal pins to ensure oscillation at the correct frequency. Rs is the value in kΩ of a resistor that is placed in series with the crystal between the `OSC1` pin and the crystal pin. Rs dissipates some of the power so the Max Dl crystal parameter is not exceeded. Only use the recommended C_1, C_2, and Rs values with the associated crystal part. The values in the table were used in the simulation to ensure crystal startup and to determine the worst case drive level (WC Dl). The value in the WC Dl column should not be greater than the Max Dl Crystal parameter. The WC Dl value can be used to determine if a crystal with similar parameter values but a lower Max Dl value is acceptable.

Table 21-18. Crystal Parameters

MFG	MFG Part#	Holder	PKG Size (mm x mm)	Freq (MHz)	Crystal Spec (Tolerance / Stability)	Crystal Parameters						Recommended Component Values			WC DI (µW)
						Typical Values			Max Values						
						C0 (pF)	C1 (fF)	L1 (mH)	ESR (Ω)	Max DI (µW)	C_L (pf)	C_1 (pF)	C_2 (pF)	Rs (kΩ)	
NDK	NX8045GB-4.000M-STD-CJL-5	NX8045GB	8 x 4.5	4	30/50 ppm	1.00	2.70	598.10	300	500	8	12	12	0	132
FOX	FQ1045A-4	2-SMD	10 x 4.5	4	30/30 ppm	1.18	4.05	396.00	150	500	10	14	14	0	103
NDK	NX8045GB-5.000M-STD-CSF-4	NX8045GB	8 x 4.5	5	30/50 ppm	1.00	2.80	356.50	250	500	8	12	12	0	164
NDK	NX8045GB-6.000M-STD-CSF-4	NX8045GB	8 x 4.5	6	30/50 ppm	1.30	4.10	173.20	250	500	8	12	12	0	214
FOX	FQ1045A-6	2-SMD	10 x 4.5	6	30/30 ppm	1.37	6.26	112.30	150	500	10	14	14	0	209
NDK	NX8045GB-8.000M-STD-CSF-6	NX8045GB	8 x 4.5	8	30/50 ppm	1.00	2.80	139.30	200	500	8	12	12	0	277
FOX	FQ7050B-8	4-SMD	7 x 5	8	30/30 ppm	1.95	6.69	59.10	80	500	10	14	14	0	217
ECS	ECS-80-16-28A-TR	HC49/US	12.5 x 4.85	8	50/30 ppm	1.82	4.90	85.70	80	500	16	24	24	0	298
Abracon	AABMM-12.0000MHz-10-D-1-X-T	ABMM	7.2 x 5.2	12	10/20 ppm	2.37	8.85	20.5	50	500	10	12	12	2.0[a]	124
NDK	NX3225GA-12.000MHZ-STD-CRG-2	NX3225GA	3.2 x 2.5	12	20/30 ppm	0.70	2.20	81.00	100	200	8	12	12	2.5	147
NDK	NX5032GA-12.000MHZ-LN-CD-1	NX5032GA	5 x 3.2	12	30/50 ppm	0.93	3.12	56.40	120	500	8	12	12	0	362
FOX	FQ5032B-12	4-SMD	5 x 3.2	12	30/30 ppm	1.16	4.16	42.30	80	500	10	14	14	0	370
Abracon	AABMM-16.0000MHz-10-D-1-X-T	ABMM	7.2 x 5.2	16	10/20 ppm	3.00	11.00	9.30	50	500	10	12	12	2.0[a]	143
Ecliptek	ECX-6595-16.000M	HC-49/UP	13.3 x 4.85	16	15/30 ppm	3.00	12.7	8.1	50	1000	10	12	12	2.0[a]	139
NDK	NX3225GA-16.000MHZ-STD-CRG-2	NX3225GA	3.2 x 2.5	16	20/30 ppm	1.00	2.90	33.90	80	200	8	12	12	2	188
NDK	NX5032GA-16.000MHZ-LN-CD-1	NX5032GA	5 x 3.2	16	30/50ppm	1.02	3.82	25.90	120[b]	500	8	10	10	0	437

Table 21-18. Crystal Parameters *(continued)*

MFG	MFG Part#	Holder	PKG Size (mm x mm)	Freq (MHz)	Crystal Spec (Tolerance / Stability)	Crystal Parameters						Recommended Component Values			WC Dl (µW)
						Typical Values			Max Values						
						C0 (pF)	C1 (fF)	L1 (mH)	ESR (Ω)	Max Dl (µW)	C_L (pf)	C_1 (pF)	C_2 (pF)	Rs (kΩ)	
ECS	ECS-160-9-42-CKM-TR	ECX-42	4 x 2.5	16	10/10 ppm	1.47	3.90	25.84	60	300	9	12	12	0.5	289
Abracon	AABMM-25.0000MHz-10-D-1-X-T	ABMM	7.2 x 5.2	25	10/20 ppm	3.00	11.00	3.70	50	500	10	12	12	2.0[a]	158
Ecliptek	ECX-6593-25.000M	HC-49/UP	13.3 x 4.85	25	15/30 ppm	3.00	12.8	3.2	40	1000	10	12	12	1.5[a]	159
NDK	NX3225GA-25.000MHZ-STD-CRG-2	NX3225GA	3.2 x 2.5	25	20/30 ppm	1.10	4.70	8.70	50	200	8	12	12	2	181
NDK	NX5032GA-25.000MHZ-LD-CD-1	NX5032GA	5 x 3.2	25	30/50 ppm	1.3	5.1	7.1	70	500	8	10	10	1.0[a]	216
												12	12	0.75[c]	269
AURIS	Q-25.000M-HC3225/4-F-30-30-E-12-TR	HC3225/4	3.2 x 2.5	25	30/30 ppm	1.58	5.01	8.34	50	500	12	16	16	1	331
FOX	FQ5032B-25	4-SMD	5 x 3.2	25	30/30 ppm	1.69	7.92	5.13	50	500	10	14	14	0.5	433
TXC	7A2570018	NX5032GA	5 x 3.2	25	20/25 ppm	2.0	6.7	6.1	30	350	10	12	12	2.0[c]	124

a. R_S values as low as 0 Ohms can be used. Using a lower R_S value will result in the WC DL to increase towards the Max DL of the crystal.

b. Although this ESR value is outside of the recommended crystal ESR maximum for this frequency, this crystal has been simulated to confirm proper operation and is valid for use with this device.

c. R_S values as low as 500 Ohms can be used. Using a lower R_S value will result in the WC DL to increase towards the Max DL of the crystal.

Table 21-19. Supported MOSC Crystal Frequencies[a]

Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL
0x00-0x5	reserved	
0x06	4 MHz	reserved
0x07	4.096 MHz	reserved
0x08	4.9152 MHz	reserved
0x09	5 MHz (USB)	
0x0A	5.12 MHz	
0x0B	6 MHz (USB)	
0x0C	6.144 MHz	
0x0D	7.3728 MHz	
0x0E	8 MHz (USB)	
0x0F	8.192 MHz	
0x10	10.0 MHz (USB)	

Table 21-19. Supported MOSC Crystal Frequencies *(continued)*

Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL
0x11	12.0 MHz (USB)	
0x12	12.288 MHz	
0x13	13.56 MHz	
0x14	14.31818 MHz	
0x15	16.0 MHz (reset value)(USB)	
0x16	16.384 MHz	
0x17	18.0 MHz (USB)	
0x18	20.0 MHz (USB)	
0x19	24.0 MHz (USB)	
0x1A	25.0 MHz (USB)	

a. Frequencies that may be used with the USB interface are indicated in the table.

21.9.5 System Clock Specification with ADC Operation

Table 21-20. System Clock Characteristics with ADC Operation

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{sysadc}	System clock frequency when the ADC module is operating (when PLL is bypassed).[a]	15.9952	16	16.0048	MHz

a. Clock frequency (plus jitter) must be stable inside specified range. ADC can be clocked from the PLL, directly from an external clock source, or from the PIOSC, as long as frequency absolute precision is inside specified range.

21.9.6 System Clock Specification with USB Operation

Table 21-21. System Clock Characteristics with USB Operation

Parameter	Parameter Name	Min	Nom	Max	Unit
F_{sysusb}	System clock frequency when the USB module is operating (note that MOSC must be the clock source, either with or without using the PLL)	20	-	-	MHz

21.10 Sleep Modes

Table 21-22. Sleep Modes AC Characteristics[a]

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
D1	T_{WAKE_S}	Time to wake from interrupt in sleep mode[b]	-	-	2	system clocks
	T_{WAKE_DS}	Time to wake from interrupt in deep-sleep mode, using PIOSC for both Run mode and Deep-sleep mode[b c]	-	1.25	-	µs
		Time to wake from interrupt in deep-sleep mode, using PIOSC for Run mode and LFIOSC for Deep-sleep mode[b c]	-	350	-	µs
D2	$T_{WAKE\_PLL\_DS}$	Time to wake from interrupt in deep-sleep mode when using the PLL[b]	-	-	T_{READY}	ms

a. Values in this table assume the LFIOSC is the clock source during sleep or deep-sleep mode.

b. Specified from registering the interrupt to first instruction.

c. If the main oscillator is used for run mode, add the main oscillator startup time, T_{START}.

Table 21-23. Time to Wake with Respect to Low-Power Modes[ab]

Mode	Run Mode Clock/Frequency	Sleep/Deep-Sleep Mode Clock/Frequency	FLASHPM	SRAMPM	Time to Wake		Unit
					Min	Max	
Sleep	MOSC, PLL on - 80MHz	MOSC, PLL on - 80MHz	0x0	0x0	0.28	0.30	µs
				0x1	33.57	35.00	µs
				0x3	33.75	35.05	µs
			0x2	0x0	105.02	109.23	µs
				0x1	137.85	143.93	µs
				0x3	138.06	143.86	µs

Table 21-23. Time to Wake with Respect to Low-Power Modes *(continued)*

Mode	Run Mode Clock/Frequency	Sleep/Deep-Sleep Mode Clock/Frequency	FLASHPM	SRAMPM	Time to Wake		Unit
					Min	Max	
Deep-Sleep	MOSC, PLL on - 80MHz	PIOSC - 16MHz	0x0	0x0	2.47	2.60	μs
				0x1	35.31	36.35	μs
				0x3	35.40	36.76	μs
			0x2	0x0	107.05	111.54	μs
				0x1	139.34	145.64	μs
				0x3	140.41	145.53	μs
	PIOSC - 16MHz	PIOSC - 16MHz	0x0	0x0	2.47	2.61	μs
				0x1	35.25	36.65	μs
				0x3	35.38	36.79	μs
			0x2	0x0	107.43	111.52	μs
				0x1	139.83	145.85	μs
				0x3	139.35	145.54	μs
	PIOSC - 16MHz	LFIOSC, PIOSC off[c] - 30kHz	0x0	0x0	415.06	728.38	μs
				0x1	436.60	740.88	μs
				0x3	433.80	755.32	μs
			0x2	0x0	503.73	812.82	μs
				0x1	537.72	846.23	μs
				0x3	536.10	839.25	μs
	MOSC, PLL on - 80MHz	LFIOSC, PIOSC off[c] - 30kHz	0x0	0x0	18.95	19.55	ms
				0x1	18.94	19.54	ms
				0x3	18.95	19.53	ms
			0x2	0x0	18.95	19.54	ms
				0x1	18.94	19.53	ms
				0x3	18.95	19.54	ms

a. Time from wake event to first instruction of code execution.

b. If the LDO voltage is adjusted, it will take an extra 4 us to wake up from Sleep or Deep-sleep mode.

c. PIOSC is turned off by setting the `PIOSCPD` bit in the **DSLPCLKCFG** register.

21.11 Flash Memory and EEPROM

Table 21-24. Flash Memory Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
PE_{CYC}	Number of program/erase cycles before failure[a]	100,000	-	-	cycles
T_{RET}	Data retention, -40˚C to +85˚C	20	-	-	years
T_{PROG64}	Program time for double-word-aligned 64 bits of data[b]	30	50	300	µs
T_{ERASE}	Page erase time, <1k cycles	-	8	15	ms
	Page erase time, 10k cycles	-	15	40	ms
	Page erase time, 100k cycles	-	75	500	ms
T_{ME}	Mass erase time, <1k cycles	-	10	25	ms
	Mass erase time, 10k cycles	-	20	70	ms
	Mass erase time, 100k cycles	-	300	2500	ms

a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

b. If programming fewer than 64 bits of data, the programming time is the same. For example, if only 32 bits of data need to be programmed, the other 32 bits are masked off.

Table 21-25. EEPROM Characteristics[a]

Parameter	Parameter Name	Min	Nom	Max	Unit
EPE_{CYC}[b]	Number of mass program/erase cycles of a single word before failure[c]	500,000	-	-	cycles
ET_{RET}	Data retention, -40˚C to +85˚C	20	-	-	years
ET_{PROG}	Program time for 32 bits of data - space available	-	110	600	µs
	Program time for 32 bits of data - requires a copy to the copy buffer, copy buffer has space and less than 10% of EEPROM endurance used	-	30	-	ms
	Program time for 32 bits of data - requires a copy to the copy buffer, copy buffer has space and greater than 90% of EEPROM endurance used	-	-	900	ms
	Program time for 32 bits of data - requires a copy to the copy buffer, copy buffer requires an erase and less than 10% of EEPROM endurance used	-	60	-	ms
	Program time for 32 bits of data - requires a copy to the copy buffer, copy buffer requires an erase and greater than 90% of EEPROM endurance used	-	-	1800	ms
ET_{READ}	Read access time	-	4	-	system clocks
ET_{ME}	Mass erase time, <1k cycles	-	8	15	ms
	Mass erase time, 10k cycles	-	15	40	ms
	Mass erase time, 100k cycles	-	75	500	ms

a. Because the EEPROM operates as a background task and does not prevent the CPU from executing from Flash memory, the operation will complete within the maximum time specified provided the EEPROM operation is not stalled by a Flash memory program or erase operation.

b. One word can be written more than 500K times, but these writes impact the endurance of the words in the meta-block that the word is within. Different words can be written such that any or all words can be written more than 500K times when write counts per word stay about the same. See the section called “Endurance” on page 471 for more information.

c. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

21.12 Input/Output Pin Characteristics

21.12.1 GPIO Module Characteristics

Note: All GPIO signals are 5-V tolerant when configured as inputs except for PD4, PD5, PB0 and PB1, which are limited to 3.6 V. See "Signal Description" on page 581 for more information on GPIO configuration.

Note: GPIO pads are tolerant to 5-V digital inputs without creating reliability issues, as long as the supply voltage, VDD, is present. There are limitations to how long a 5-V input can be present on any given I/O pad if VDD is not present. Not meeting these conditions will affect reliability of the device and affect the GPIO characteristics specifications.

- If the voltage applied to a GPIO pad is in the high voltage range (5V +/- 10%) while VDD is not present, such condition should be allowed for a maximum of 10,000 hours at 27°C or 5,000 hours at 85°C, over the lifetime of the device.
- If the voltage applied to a GPIO pad is in the normal voltage range (3.3V +/- 10%) while VDD is not present or if the voltage applied is in the high voltage range (5V +/- 10%) while VDD is present, there are no constraints on the lifetime of the device.

Table 21-26. GPIO Module Characteristics[a]

Parameter	Parameter Name	Min	Nom	Max	Unit
C_{GPIO}	GPIO Digital Input Capacitance	-	8	-	pF
R_{GPIOPU}	GPIO internal pull-up resistor	13	20	30	kΩ
R_{GPIOPD}	GPIO internal pull-down resistor	13	20	35	kΩ
I_{LKG+}	GPIO input leakage current, 0 V ≤ V_{IN} ≤ V_{DD} GPIO pins[b]	-	-	1.0	µA
	GPIO input leakage current, 0 V < V_{IN} ≤ V_{DD}, GPIO pins configured as ADC or analog comparator inputs	-	-	2.0	µA
T_{GPIOR}	GPIO rise time, 2-mA drive[c]	-	14.2	16.1	ns
	GPIO rise time, 4-mA drive[c]		11.9	15.5	ns
	GPIO rise time, 8-mA drive[c]		8.1	11.2	ns
	GPIO rise time, 8-mA drive with slew rate control[c]		9.5	11.8	ns
T_{GPIOF}	GPIO fall time, 2-mA drive[d]	-	25.2	29.4	ns
	GPIO fall time, 4-mA drive[d]		13.3	16.8	ns
	GPIO fall time, 8-mA drive[d]		8.6	11.2	ns
	GPIO fall time, 8-mA drive with slew rate control[d]		11.3	12.9	ns

[a] V_{DD} must be within the range specified in Table 21-5 on page 1124.

[b] The leakage current is measured with V_{IN} applied to the corresponding pin(s). The leakage of digital port pins is measured individually. The port pin is configured as an input and the pull-up/pull-down resistor is disabled.

[c] Time measured from 20% to 80% of V_{DD}.

[d] Time measured from 80% to 20% of V_{DD}.

21.12.2 Types of I/O Pins and ESD Protection

With respect to ESD and leakage current, three types of I/O pins exist on the device: Power I/O pins, I/O pins with fail-safe ESD protection (GPIOs other than PD4 and PD5) and I/O pins with non-fail-safe ESD protection (any non-power, non-GPIO (other than PD4 and PD5) and pins). This section covers I/O pins with fail-safe ESD protection and I/O pins with non-fail-safe ESD protection. Power I/O pin voltage and current limitations are specified in “Recommended Operating Conditions” on page 1124.

21.12.2.1 Fail-Safe Pins

GPIOs other than PD4 and PD5 and I/O pins for the USB PHY use ESD protection as shown in Figure 21-15 on page 1148.

An unpowered device cannot be parasitically powered through any of these pins. This ESD protection prevents a direct path between these I/O pads and any power supply rails in the device. GPIO pad input voltages should be kept inside the maximum ratings specified in Table 21-1 on page 1122 to ensure current leakage and current injections are within acceptable range. Current leakages and current injection for these pins are specified in Table 21-26 on page 1147.

Figure 21-15 on page 1148 shows a diagram of the ESD protection on fail-safe pins.

Some GPIOs when configured as inputs require a strong pull-up resistor to maintain a threshold above the minimum value of VIH during power-on. See Table 21-28 on page 1149.

Figure 21-15. ESD Protection on Fail-Safe Pins

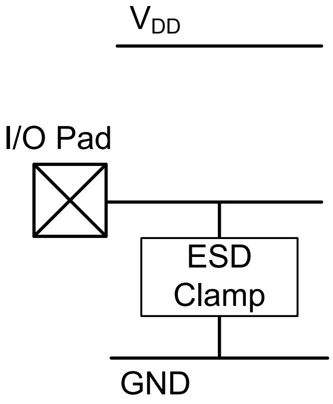


Table 21-27. Pad Voltage/Current Characteristics for Fail-Safe Pins[a]

Parameter	Parameter Name	Min	Nom	Max	Unit
I_{LKG+}	GPIO input leakage current, $V_{DD}< V_{IN} \le 4.5$ V[bb]	-	-	700	µA
	GPIO input leakage current, 4.5 V < $V_{IN} \le 5.5$ V[bc]	-	-	100	µA
I_{LKG-}	GPIO input leakage current, $V_{IN} < -0.3$ V[bd]	-	-	-[e]	µA
	GPIO input leakage current, -0.3 V ≤ V_{IN} < 0 V[b]	-	-	10	µA
I_{INJ+}	DC injection current, $V_{DD} < V_{IN} \le 5.5$ V[fg]	-	-	I_{LKG+}	µA
I_{INJ-}	DC injection current, $V_{IN} \le 0$ V[g]	-	-	0.5	mA

a. VIN must be within the range specified in Table 21-1 on page 1122.

b. To protect internal circuitry from over-voltage, the GPIOs have an internal voltage clamp that limits internal swings to V_{DD} without affecting swing at the I/O pad. This internal clamp starts turning on while $V_{DD} < V_{IN} < 4.5$ V and causes a somewhat larger (but bounded) current draw. To save power, static input voltages between V_{DD} and 4.5 V should be avoided.

c. Leakage current above maximum voltage (V_{IN} = 5.5V) is not guaranteed, this condition is not allowed and can result in permanent damage to the device.

d. Leakage outside the minimum range (-0.3V) is unbounded and must be limited to IINJ- using an external resistor.

e. In this case, I_{LKG-} is unbounded and must be limited to I_{INJ-} using an external resistor.

f. Current injection is internally bounded for GPIOs, and maximum current into the pin is given by ILKG+ for $V_{DD} < V_{IN} < 5.5$ V.

g. If the I/O pad is not voltage limited, it should be current limited (to IINJ+ and IINJ-) if there is any possibility of the pad voltage exceeding the VIO limits (including transient behavior during supply ramp up, or at any time when the part is unpowered).

Table 21-28. Fail-Safe GPIOs that Require an External Pull-up

GPIO	Pin	Pull-Up Resistor Value	Unit
PB0	45	1k ≤ R ≤ 10k	Ω
PB1	46	1k ≤ R ≤ 10k	Ω
PE3	6	1k ≤ R ≤ 10k	Ω

21.12.2.2 Non-Fail-Safe Pins

The Main Oscillator (MOSC) crystal connection pins and GPIO pins PD4 and PD5 have ESD protection as shown in Figure 21-16 on page 1149. These pins have a potential path between the I/O pad and an internal power rail if either one of the ESD diodes is accidentally forward biased. The voltage and current of these pins should follow the specifications in Table 21-29 on page 1149 to prevent potential damage to the device. In addition to the specifications outlined in Table 21-29 on page 1149, it is recommended that the ADC external reference specifications in Table 21-30 on page 1151 be adhered to in order to prevent any gain error.

Figure 21-16 on page 1149 shows a diagram of the ESD protection on non-fail-safe pins.

Figure 21-16. ESD Protection on Non-Fail-Safe Pins

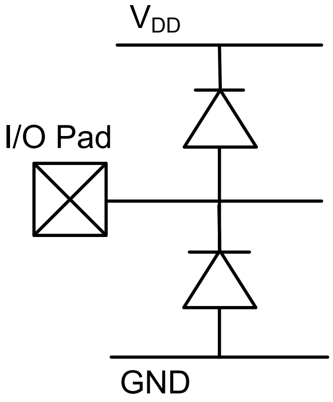


Table 21-29. Non-Fail-Safe I/O Pad Voltage/Current Characteristics[abcd]

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{IO}	IO pad voltage limits	-0.3	V_{DD}	V_{DD}+0.3	V
I_{LKG+}	Positive IO leakage for V_{IO} Max[ef]	-	-	10	µA
I_{LKG-}	Negative IO leakage for V_{IO} Min[ef]	-	-	10	µA
I_{INJ+}	Max positive injection[g]	-	-	2	mA
I_{INJ-}	Max negative injection if not voltage protected[g]	-	-	-0.5	mA

a. V_{IN} must be within the range specified in Table 21-1 on page 1122. Leakage current outside of this maximum voltage is not guaranteed and can result in permanent damage of the device.

b. VDD must be within the range specified in Table 21-5 on page 1124.

c. To avoid potential damage to the part, either the voltage or current on the ESD-protected, non-Power, non-Hibernate/XOSC input/outputs should be limited externally as shown in this table.

d. I/O pads should be protected if at any point the IO voltage has a possibility of going outside the limits shown in the table. If the part is unpowered, the IO pad Voltage or Current must be limited (as shown in this table) to avoid powering the part through the IO pad, causing potential irreversible damage.

e. This value applies to an I/O pin that is voltage-protected within the Min and Max V_{IO} ratings. Leakage outside the specified voltage range is unbounded and must be limited to I_{INJ-} using an external resistor.

f. MIN and MAX leakage current for the case when the I/O is voltage-protected to VIO Min or VIO Max.

g. If an I/O pin is not voltage-limited, it should be current-limited (to I_{INJ+} and I_{INJ-}) if there is any possibility of the pad voltage exceeding the V_{IO} limits (including transient behavior during supply ramp up, or at any time when the part is unpowered).

21.13 Analog-to-Digital Converter (ADC)

Table 21-30. ADC Electrical Characteristics[a][b]

Parameter	Parameter Name	Min	Nom	Max	Unit
POWER SUPPLY REQUIREMENTS					
V_{DDA}	ADC supply voltage	2.97	3.3	3.63	V
GNDA	ADC ground voltage	-	0	-	V
VDDA / GNDA VOLTAGE REFERENCE					
C_{REF}	Voltage reference decoupling capacitance	-	1.0 // 0.01[c]	-	µF
ANALOG INPUT					
V_{ADCIN}	Single-ended, full- scale analog input voltage, internal reference[d][e]	0	-	V_{DDA}	V
	Differential, full-scale analog input voltage, internal reference[d][f]	$-V_{DDA}$	-	V_{DDA}	V
VIN_{CM}	Input common mode voltage, differential mode[g]	-	-	(VREFP + VREFN) / 2 ± 25	mV
I_L	ADC input leakage current[h]	-	-	2.0	µA
R_{ADC}	ADC equivalent input resistance[h]	-	-	2.5	kΩ
C_{ADC}	ADC equivalent input capacitance[h]	-	-	10	pF
R_S	Analog source resistance[h]	-	-	500	Ω
SAMPLING DYNAMICS					
F_{ADC}	ADC conversion clock frequency[i]	-	16	-	MHz
F_{CONV}	ADC conversion rate		1		Msps
T_S	ADC sample time	-	250	-	ns
T_C	ADC conversion time[j]		1		µs
T_{LT}	Latency from trigger to start of conversion	-	2	-	ADC clocks
SYSTEM PERFORMANCE when using internal reference					
N	Resolution		12		bits
INL	Integral nonlinearity error, over full input range	-	±1.5	±3.0	LSB
DNL	Differential nonlinearity error, over full input range	-	±0.8	+2.0/-1.0[k]	LSB
E_O	Offset error	-	±5.0	±15.0	LSB
E_G	Gain error[l]	-	±10.0	±30.0	LSB
E_T	Total unadjusted error, over full input range[m]	-	±10.0	±30.0	LSB
DYNAMIC CHARACTERISTICS[n][o]					
SNR_D	Signal-to-noise-ratio, Differential input, V_{ADCIN}: -20dB FS, 1KHz [p]	70	72	-	dB
SDR_D	Signal-to-distortion ratio, Differential input, V_{ADCIN}: -3dB FS, 1KHz[p][q][r]	72	75	-	dB
$SNDR_D$	Signal-to-Noise+Distortion ratio, Differential input, V_{ADCIN}: -3dB FS, 1KHz[p][s][t]	68	70	-	dB
SNR_S	Signal-to-noise-ratio, Single-ended input, V_{ADCIN}: -20dB FS, 1KHz [u]	60	65	-	dB

Table 21-30. ADC Electrical Characteristics *(continued)*

Parameter	Parameter Name	Min	Nom	Max	Unit
SDR_S	Signal-to-distortion ratio, Single-ended input, V_{ADCIN}: -3dB FS, 1KHz[q][r]	70	72	-	dB
$SNDR_S$	Signal-to-Noise+Distortion ratio, Single-ended input, V_{ADCIN}: -3dB FS, 1KHz[s][t][u]	60	63	-	dB
TEMPERATURE SENSOR					
V_{TSENS}	Temperature sensor voltage, junction temperature 25 °C	-	1.633	-	V
S_{TSENS}	Temperature sensor slope	-	-13.3	-	mV/°C
E_{TSENS}	Temperature sensor accuracy[v]	-	-	±5	°C

a. V_{REF+}= 3.3V, F_{ADC}=16 MHz unless otherwise noted.

b. Best design practices suggest that static or quiet digital I/O signals be configured adjacent to sensitive analog inputs to reduce capacitive coupling and cross talk. Analog signals configured adjacent to ADC input channels should meet the same source resistance and bandwidth limitations that apply to the ADC input signals.

c. Two capacitors in parallel.

d. Internal reference is connected directly between V_{DDA} and GNDA (VREFi = V_{DDA} - GNDA). In this mode, E_O, E_G, E_T, and dynamic specifications are adversely affected due to internal voltage drop and noise on V_{DDA} and GNDA.

e. $V_{ADCIN} = V_{INP} - V_{INN}$

f. With signal common mode as $V_{DDA}/2$.

g. This parameter is defined as the average of the differential inputs.

h. As shown in Figure 21-17 on page 1153, R_{ADC} is the total equivalent resistance in the input line all the way up to the sampling node at the input of the ADC.

i. See "System Clock Specification with ADC Operation" on page 1143 for full ADC clock frequency specification.

j. ADC conversion time (Tc) includes the ADC sample time (Ts).

k. 12-bit DNL

l. Gain error is measured at max code after compensating for offset. Gain error is equivalent to "Full Scale Error." It can be given in % of slope error, or in LSB, as done here.

m. Total Unadjusted Error is the maximum error at any one code versus the ideal ADC curve. It includes all other errors (offset error, gain error and INL) at any given ADC code.

n. A low-noise environment is assumed in order to obtain values close to spec. The board must have good ground isolation between analog and digital grounds and a clean reference voltage. The input signal must be band-limited to Nyquist bandwidth. No anti-aliasing filter is provided internally.

o. ADC dynamic characteristics are measured using low-noise board design, with low-noise reference voltage (< -74dB noise level in signal BW) and low-noise analog supply voltage. Board noise and ground bouncing couple into the ADC and affect dynamic characteristics. Clean external reference must be used to achieve shown specs.

p. Differential signal with correct common mode, applied between two ADC inputs.

q. SDR = -THD in dB.

r. For higher frequency inputs, degradation in SDR should be expected.

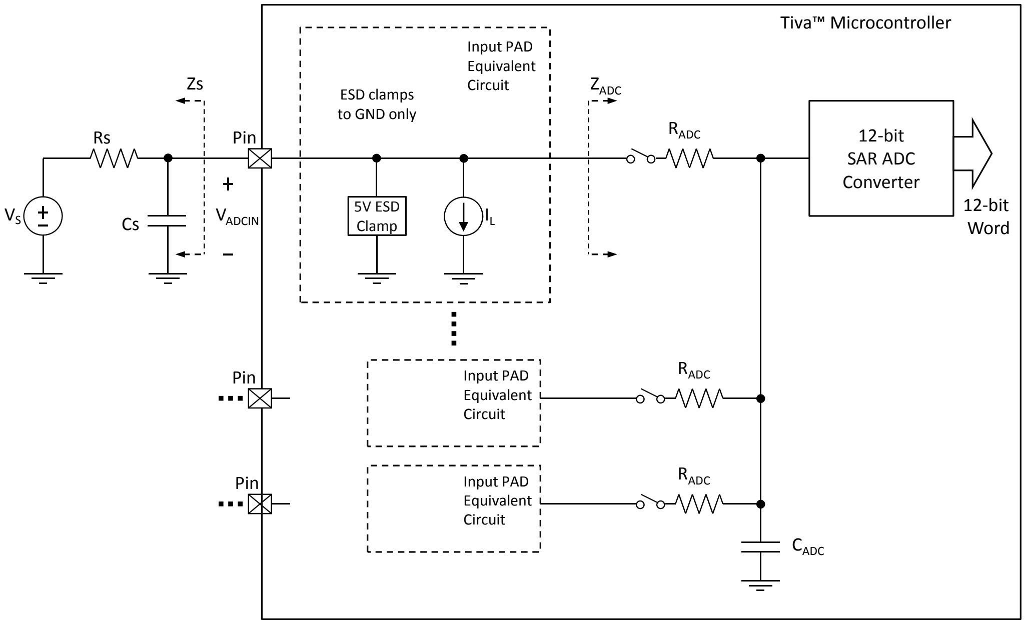
s. SNDR = S/(N+D) = SINAD (in dB)

t. Effective number of bits (ENOB) can be calculated from SNDR: ENOB = (SNDR - 1.76) / 6.02.

u. Single-ended inputs are more sensitive to board and trace noise than differential inputs; SNR and SNDR measurements on single-ended inputs are highly dependent on how clean the test set-up is. If the input signal is not well-isolated on the board, higher noise than specified could potentially be seen at the ADC output.

v. Note that this parameter does not include ADC error.

Figure 21-17. ADC Input Equivalency Diagram



21.14 Synchronous Serial Interface (SSI)

Table 21-31. SSI Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	T_{CLK_PER}	SSIClk cycle time, as master[a]	40	-	-	ns
		SSIClk cycle time, as slave[b]	150	-	-	ns
S2	T_{CLK_HIGH}	SSIClk high time, as master	20	-	-	ns
		SSIClk high time, as slave	75	-	-	ns
S3	T_{CLK_LOW}	SSIClk low time, as master	20	-	-	ns
		SSIClk low time, as slave	75	-	-	ns
S4	T_{CLKR}	SSIClk rise time[c]	1.25	-	-	ns
S5	T_{CLKF}	SSIClk fall time[c]	1.25	-	-	ns
S6	T_{TXDMOV}	Master Mode: Master Tx Data Output (to slave) Valid Time from edge of SSIClk	-	-	15.7	ns
S7	T_{TXDMOH}	Master Mode: Master Tx Data Output (to slave) Hold Time from next SSIClk	0.31	-	-	ns
S8	T_{RXDMS}	Master Mode: Master Rx Data In (from slave) setup time	17.15	-	-	ns
S9	T_{RXDMH}	Master Mode: Master Rx Data In (from slave) hold time	0	-	-	ns
S10	T_{TXDSOV}	Slave Mode: Master Tx Data Output (to Master) Valid Time from edge of SSIClk	-	-	77.74[d]	ns
S11	T_{TXDSOH}	Slave Mode: Slave Tx Data Output (to Master) Hold Time from next SSIClk	55.5[e]	-	-	ns
S12	T_{RXDSSU}	Slave Mode: Rx Data In (from master) setup time	0	-	-	ns
S13	T_{RXDSH}	Slave Mode: Rx Data In (from master) hold time	51.55[f]	-	-	ns

[a] In master mode, the system clock must be at least twice as fast as the SSIClk.

[b] In slave mode, the system clock must be at least 12 times faster than the SSIClk.

[c] Note that the delays shown are using 8-mA drive strength.

[d] This MAX value is for the minimum T_{SYSCLK} (12.5 ns). To find the MAX T_{TXDSOV} value for a larger T_{SYSCLK}, use the equation: $4*T_{SYSCLK}+27.74$.

[e] This MIN value is for the minimum slave mode T_{SYSCLK} (12.5 ns). To find the MIN T_{TXDSOH} value for a larger T_{SYSCLK}, use the equation: $4*T_{SYSCLK}+5.50$.

[f] This MIN value is for the minimum slave mode T_{SYSCLK} (12.5 ns). To find the MIN T_{RXDSH} value for a larger T_{SYSCLK}, use the equation: $4*T_{SYSCLK}+1.55$.

Figure 21-18. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

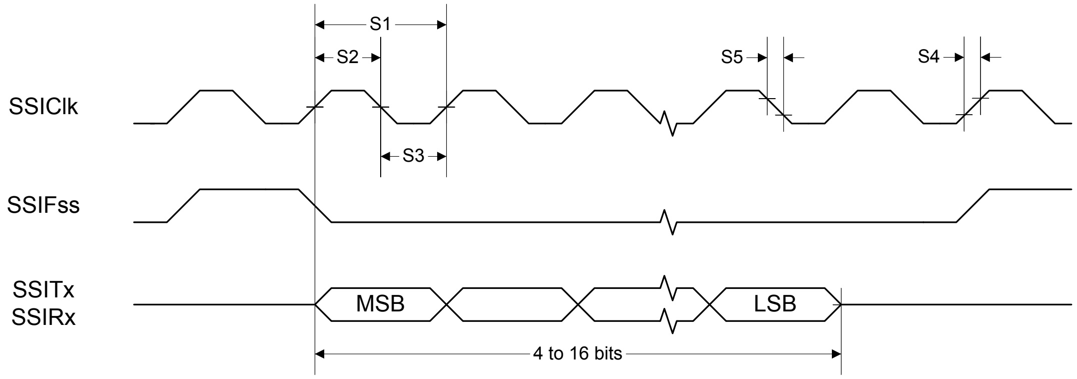


Figure 21-19. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

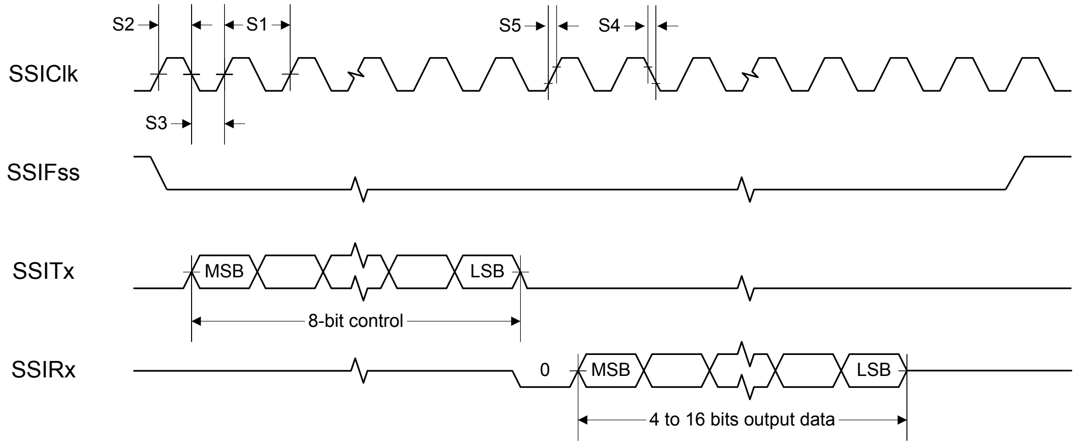


Figure 21-20. Master Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1

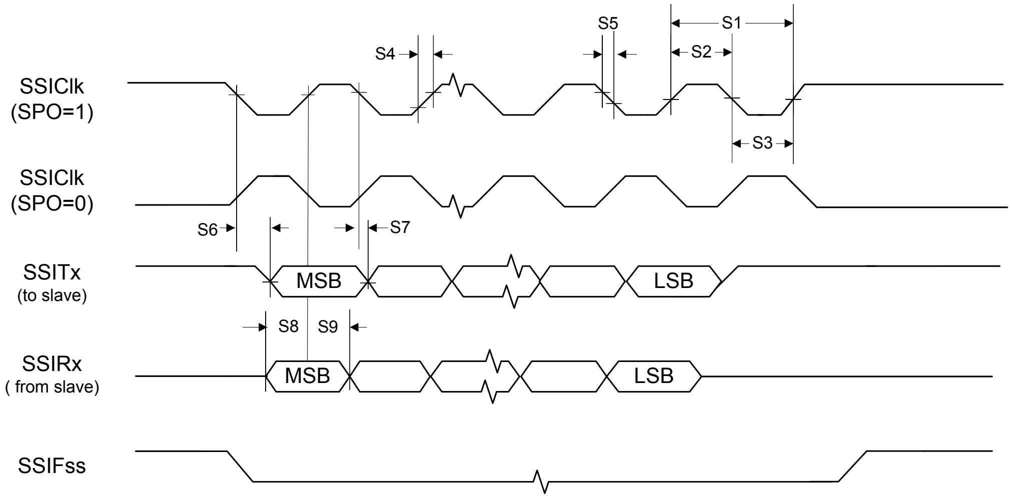
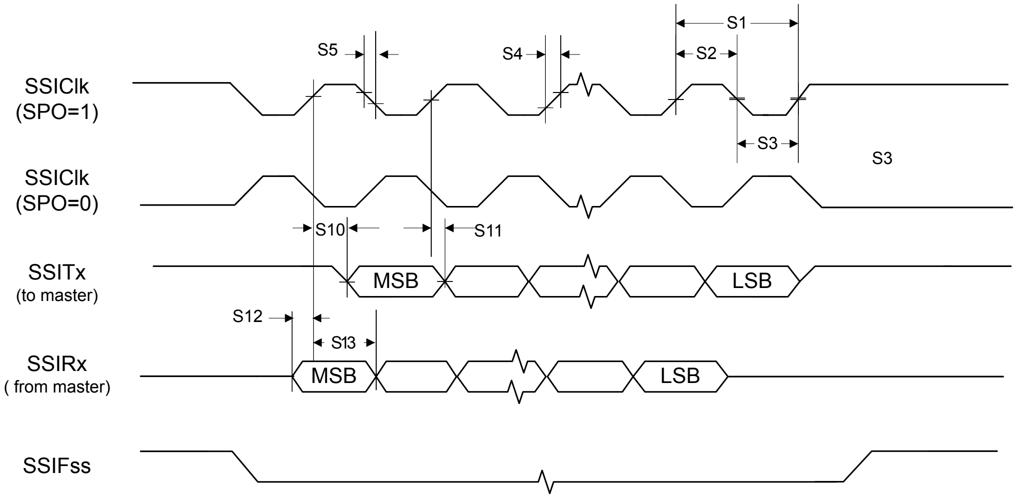


Figure 21-21. Slave Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1



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

Table 21-32. I^2C Characteristics

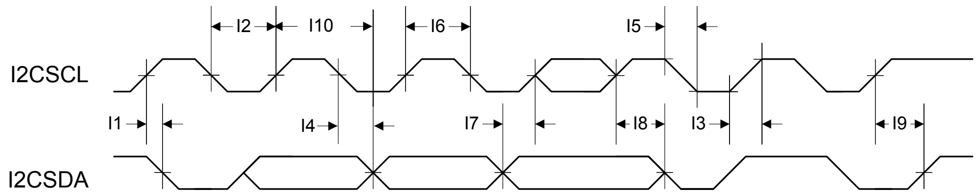
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
I1[a]	T_{SCH}	Start condition hold time	36	-	-	system clocks
I2[a]	T_{LP}	Clock Low period	36	-	-	system clocks
I3[b]	T_{SRT}	I2CSCL/I2CSDA rise time (V_{IL} =0.5 V to V_{IH} =2.4 V)	-	-	(see note b)	ns
I4	T_{DH}	Data hold time (slave)	-	2	-	system clocks
		Data hold time (master)	-	7	-	system clocks
I5[c]	T_{SFT}	I2CSCL/I2CSDA fall time (V_{IH} =2.4 V to V_{IL} =0.5 V)	-	9	10	ns
I6[a]	T_{HT}	Clock High time	24	-	-	system clocks
I7	T_{DS}	Data setup time	18	-	-	system clocks
I8[a]	T_{SCSR}	Start condition setup time (for repeated start condition only)	36	-	-	system clocks
I9[a]	T_{SCS}	Stop condition setup time	24	-	-	system clocks
I10	T_{DV}	Data Valid (slave)	-	2	-	system clocks
		Data Valid (master)	-	(6 * (1 + TPR)) + 1	-	system clocks

a. Values depend on the value programmed into the TPR bit in the **I^2C Master Timer Period (I2CMTPR)** register; a TPR programmed for the maximum I2CSCL frequency (TPR=0x2) results in a minimum output timing as shown in the table above. The I^2C interface is designed to scale the actual data transition time to move it to the middle of the I2CSCL Low period. The actual position is affected by the value programmed into the TPR; however, the numbers given in the above values are minimum values.

b. Because I2CSCL and I2CSDA operate as open-drain-type signals, which the controller can only actively drive Low, the time I2CSCL or I2CSDA takes to reach a high level depends on external signal capacitance and pull-up resistor values.

c. Specified at a nominal 50 pF load.

Figure 21-22. I^2C Timing



21.16 Universal Serial Bus (USB) Controller

The TM4C1232C3PM USB controller electrical specifications are compliant with the *Universal Serial Bus Specification Rev. 2.0* (full-speed and low-speed support). Some components of the USB system are integrated within the TM4C1232C3PM microcontroller and specific to the TM4C1232C3PM microcontroller design.

21.17 Analog Comparator

Table 21-33. Analog Comparator Characteristics[a][b]

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{INP},V_{INN}[c]	Input voltage range	GNDA	-	V_{DDA}	V
V_{CM}	Input common mode voltage range	GNDA	-	V_{DDA}	V
V_{OS}	Input offset voltage	-	±10	±50[d]	mV
I_{INP},I_{INN}	Input leakage current over full voltage range	-	-	2.0	µA
C_{MRR}	Common mode rejection ratio	-	50	-	dB
T_{RT}	Response time	-	-	1.0[e]	µs
T_{MC}	Comparator mode change to Output Valid	-	-	10	µs

a. Best design practices suggest that static or quiet digital I/O signals be configured adjacent to sensitive analog inputs to reduce capacitive coupling and cross talk.

b. To achieve best analog results, the source resistance driving the analog inputs, V_{INP} and V_{INN}, should be kept low.

c. The external voltage inputs to the Analog Comparator are designed to be highly sensitive and can be affected by external noise on the board. For this reason, V_{INP} and V_{INN} must be set to different voltage levels during idle states to ensure the analog comparator triggers are not enabled. If an internal voltage reference is used, it should be set to a mid-supply level. When operating in Sleep/Deep-Sleep modes, the Analog Comparator module external voltage inputs set to different levels (greater than the input offset voltage) to achieve minimum current draw.

d. Measured at VREF=100 mV.

e. Measured at external VREF=100 mV, input signal switching from 75 mV to 125 mV.

Table 21-34. Analog Comparator Voltage Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{HR}	Resolution in high range	-	V_{DDA}/29.4	-	V
R_{LR}	Resolution in low range	-	V_{DDA}/22.12	-	V
A_{HR}	Absolute accuracy high range	-	-	±R_{HR}/2	V
A_{LR}	Absolute accuracy low range	-	-	±R_{LR}/2	V

Table 21-35. Analog Comparator Voltage Reference Characteristics, V_{DDA} = 3.3V, EN= 1, and RNG = 0

VREF Value	V_{IREF} Min	Ideal V_{IREF}	V_{IREF} Max	Unit
0x0	0.731	0.786	0.841	V
0x1	0.843	0.898	0.953	V
0x2	0.955	1.010	1.065	V
0x3	1.067	1.122	1.178	V
0x4	1.180	1.235	1.290	V
0x5	1.292	1.347	1.402	V
0x6	1.404	1.459	1.514	V
0x7	1.516	1.571	1.627	V
0x8	1.629	1.684	1.739	V
0x9	1.741	1.796	1.851	V
0xA	1.853	1.908	1.963	V
0xB	1.965	2.020	2.076	V
0xC	2.078	2.133	2.188	V

Table 21-35. Analog Comparator Voltage Reference Characteristics, V_{DDA} = 3.3V, EN= 1, and RNG = 0 *(continued)*

VREF Value	V_{IREF} Min	Ideal V_{IREF}	V_{IREF} Max	Unit
0xD	2.190	2.245	2.300	V
0xE	2.302	2.357	2.412	V
0xF	2.414	2.469	2.525	V

Table 21-36. Analog Comparator Voltage Reference Characteristics, V_{DDA} = 3.3V, EN= 1, and RNG = 1

VREF Value	V_{IREF} Min	Ideal V_{IREF}	V_{IREF} Max	Unit
0x0	0.000	0.000	0.074	V
0x1	0.076	0.149	0.223	V
0x2	0.225	0.298	0.372	V
0x3	0.374	0.448	0.521	V
0x4	0.523	0.597	0.670	V
0x5	0.672	0.746	0.820	V
0x6	0.822	0.895	0.969	V
0x7	0.971	1.044	1.118	V
0x8	1.120	1.193	1.267	V
0x9	1.269	1.343	1.416	V
0xA	1.418	1.492	1.565	V
0xB	1.567	1.641	1.715	V
0xC	1.717	1.790	1.864	V
0xD	1.866	1.939	2.013	V
0xE	2.015	2.089	2.162	V
0xF	2.164	2.238	2.311	V

21.18 Current Consumption

Table 21-37. Current Consumption

Parameter	Parameter Name	Conditions	System Clock		Nom			Max	Unit
			Frequency	Clock Source	-40°C	25°C	85°C	85°C	
I_{DD_RUN}	Run mode (Flash loop)	V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All ON	80 MHz	MOSC with PLL	45.0	45.1	45.7	54.9	mA
			40 MHz	MOSC with PLL	31.9	32.0	32.7	40.6	mA
			16 MHz	MOSC with PLL	19.6	19.7	20.3	27.6	mA
			16 MHz	PIOSC	17.5	17.6	18.0	25.3	mA
			1 MHz	PIOSC	10.0	10.1	10.5	17.5	mA
		V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All OFF	80 MHz	MOSC with PLL	24.5	24.7	25.2	31.3	mA
			40 MHz	MOSC with PLL	19.6	19.7	20.4	25.9	mA
			16 MHz	MOSC with PLL	12.1	12.2	12.7	18.7	mA
			16 MHz	PIOSC	10.1	10.1	10.5	16.4	mA
			1 MHz	PIOSC	5.45	5.50	5.98	11.6	mA
	Run mode (SRAM loop)	V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All ON	80 MHz	MOSC with PLL	34.7	34.9	35.5	44.2	mA
			40 MHz	MOSC with PLL	22.2	22.4	22.9	30.2	mA
			16 MHz	MOSC with PLL	14.7	14.8	15.3	21.8	mA
			16 MHz	PIOSC	12.8	12.9	13.4	19.7	mA
			1 MHz	PIOSC	8.07	8.16	8.61	14.6	mA
		V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All OFF	80 MHz	MOSC with PLL	15.2	15.3	15.8	21.7	mA
			40 MHz	MOSC with PLL	10.3	10.5	10.9	16.2	mA
			16 MHz	MOSC with PLL	7.32	7.45	7.92	13.0	mA
			16 MHz	PIOSC	5.87	5.96	6.35	13.7	mA
			1 MHz	PIOSC	3.54	3.63	4.07	8.84	mA
I_{DDA}[a]	Run, Sleep and Deep-sleep mode	V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All ON	-	MOSC with PLL, PIOSC	2.71	2.71	2.71	3.97	mA
	Deep-Sleep mode		30 kHz	LFIOSC	2.54	2.54	2.54	3.68	mA
	Run, Sleep and Deep-sleep mode	V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All OFF	-	MOSC with PLL, PIOSC, LFIOSC	0.28	0.28	0.29	0.56	mA

Table 21-37. Current Consumption *(continued)*

Parameter	Parameter Name	Conditions	System Clock		Nom			Max	Unit
			Frequency	Clock Source	-40°C	25°C	85°C	85°C	
I_{DD_SLEEP}	Sleep mode (FLASHPM = 0x0)	V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All ON LDO = 1.2 V	80 MHz	MOSC with PLL	29.3	29.5	30.0	38.1	mA
			40 MHz	MOSC with PLL	19.5	19.7	20.2	27.1	mA
			16 MHz	MOSC with PLL	13.6	13.8	14.2	20.6	mA
			16 MHz	PIOSC[b]	11.7	11.8	12.2	18.5	mA
			1 MHz	PIOSC[b]	7.01	7.06	7.93	12.0	mA
		V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All OFF LDO = 1.2 V	80 MHz	MOSC with PLL	9.60	9.73	10.2	15.4	mA
			40 MHz	MOSC with PLL	7.49	7.60	8.06	13.2	mA
			16 MHz	MOSC with PLL	6.22	6.33	6.78	11.7	mA
			16 MHz	PIOSC[b]	4.28	4.35	4.77	9.52	mA
			1 MHz	PIOSC[b]	3.52	3.59	4.01	8.70	mA
	Sleep mode (FLASHPM = 0x2)	V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All ON LDO = 1.2 V	80 MHz	MOSC with PLL	28.4	28.6	29.2	37.2	mA
			40 MHz	MOSC with PLL	18.6	18.8	19.3	26.2	mA
			16 MHz	MOSC with PLL	12.7	12.9	13.3	19.7	mA
			16 MHz	PIOSC[b]	10.8	10.9	11.3	17.5	mA
			1 MHz	PIOSC[b]	7.09	7.20	7.67	13.6	mA
		V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All OFF LDO = 1.2 V	80 MHz	MOSC with PLL	8.66	8.82	9.31	14.5	mA
			40 MHz	MOSC with PLL	6.55	6.69	7.17	12.1	mA
			16 MHz	MOSC with PLL	5.27	5.41	5.89	10.7	mA
			16 MHz	PIOSC[b]	3.34	3.44	3.88	8.65	mA
			1 MHz	PIOSC[b]	2.58	2.67	3.13	7.85	mA

Table 21-37. Current Consumption *(continued)*

Parameter	Parameter Name	Conditions	System Clock		Nom			Max	Unit
			Frequency	Clock Source	-40°C	25°C	85°C	85°C	
$I_{DD\_DEEPSLEEP}$	Deep-sleep mode (FLASHPM = 0x0)	V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All ON LDO = 1.2 V	16 MHz	PIOSC	9.29	9.29	9.66	15.9	mA
			30 kHz	LFIOSC	5.10	5.10	5.48	11.2	mA
		V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All OFF LDO = 1.2 V	16 MHz	PIOSC	3.51	3.51	3.91	8.67	mA
			30 kHz	LFIOSC	2.00	2.00	2.39	7.24	mA
	Deep-sleep mode (FLASHPM = 0x2)	V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All ON LDO = 1.2 V	16 MHz	PIOSC	8.34	8.36	8.77	14.9	mA
			30 kHz	LFIOSC	4.14	4.18	4.59	10.4	mA
		V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All OFF LDO = 1.2 V	16 MHz	PIOSC	2.56	2.60	3.02	7.79	mA
			30 kHz	LFIOSC	1.04	1.07	1.49	6.48	mA

[a] The value for I_{DDA} is included in the above values for I_{DD_RUN}, I_{DD_SLEEP}, and $I_{DD\_DEEPSLEEP}$.

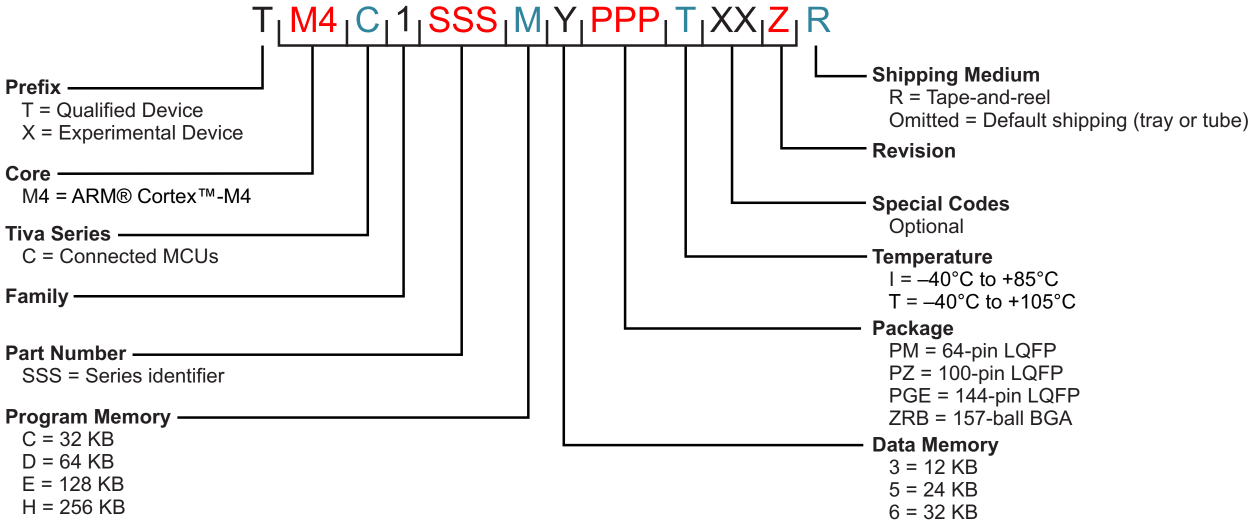
[b] Note that if the MOSC is the source of the Run-mode system clock and is powered down in Sleep mode, wake time is increased by T_{MOSC_SETTLE}.

A Package Information

A.1 Orderable Devices

The figure below defines the full set of orderable part numbers for the TM4C123x Series. See the Package Option Addendum for the complete list of valid orderable part numbers for the TM4C1232C3PM microcontroller.

Figure A-1. Key to Part Numbers



A.2 Device Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all microcontroller (MCU) devices. Each Tiva™ C Series family member has one of two prefixes: XM4C or TM4C. These prefixes represent evolutionary stages of product development from engineering prototypes (XM4C) through fully qualified production devices (TM4C).

Device development evolutionary flow:

- XM4C — Experimental device that is not necessarily representative of the final device's electrical specifications and may not use production assembly flow.
- TM4C — Production version of the silicon die that is fully qualified.

XM4C devices are shipped against the following disclaimer:

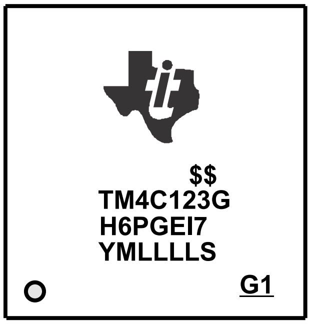
"Developmental product is intended for internal evaluation purposes."

TM4C devices have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (XM4C) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

A.3 Device Markings

The figure below shows an example of the Tiva™ microcontroller package symbolization.



This identifying number contains the following information:

- **Lines 1 and 5:** Internal tracking numbers
- **Lines 2 and 3:** Part number

  For example, TM4C123G on the second line followed by H6PGEI7 on the third line indicates orderable part number TM4C123GH6PGEI7. The silicon revision number is the last number in the part number, in this example, 7. The **DID0** register also identifies the version of the microcontroller, as shown in the table below. Combined, the MAJOR and MINOR bit fields indicate the die revision and part revision numbers.

MAJOR Bitfield Value	MINOR Bitfield Value	Die Revision	Part Revision
0x0	0x0	A0	1
0x0	0x1	A1	2
0x0	0x2	A2	3
0x0	0x3	A3	4
0x1	0x0	B0	5
0x1	0x1	B1	6
0x1	0x2	B2	7

- **Line 4:** Date code

  The first two characters on the fourth line indicate the date code, followed by internal tracking numbers. The two-digit date code YM indicates the last digit of the year, then the month. For example, a 34 for the first two digits of the fourth line indicates a date code of April 2013.

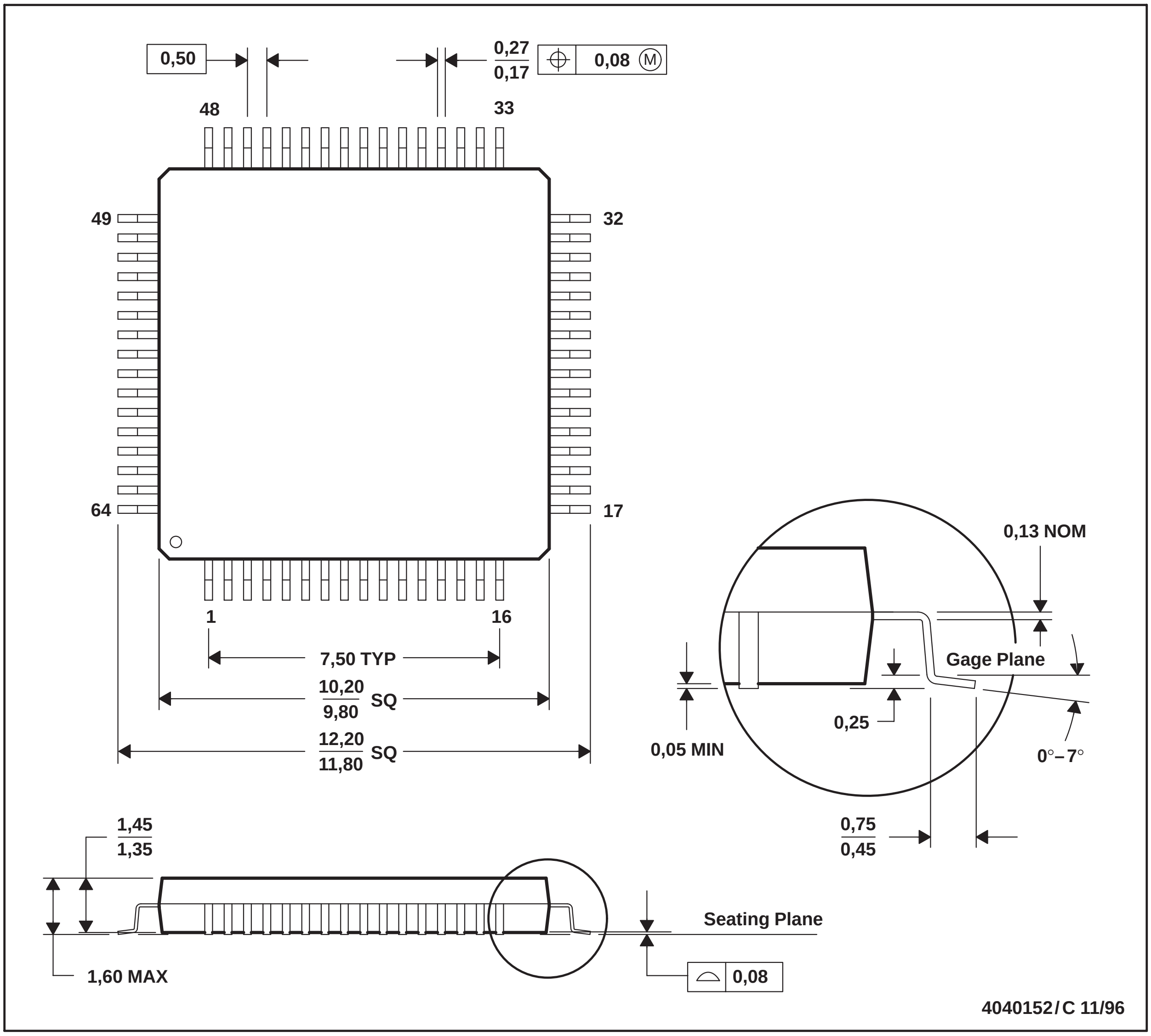
A.4 Packaging Diagram

Figure A-2. TM4C1232C3PM 64-Pin LQFP Package Diagram

MECHANICAL DATA

MTQF008A – JANUARY 1995 – REVISED DECEMBER 1996

PM (S-PQFP-G64) **PLASTIC QUAD FLATPACK**



NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS-026
D. May also be thermally enhanced plastic with leads connected to the die pads.

PACKAGING INFORMATION

Orderable part number	Status (1)	Material type (2)	Package | Pins	Package qty | Carrier	RoHS (3)	Lead finish/ Ball material (4)	MSL rating/ Peak reflow (5)	Op temp (°C)	Part marking (6)
TM4C1232C3PMI	Obsolete	Production	LQFP (PM) | 64	-	-	Call TI	Call TI	-40 to 85	TM4C1232 C3PMI
TM4C1232C3PMI7R	Obsolete	Production	LQFP (PM) | 64	-	-	Call TI	Call TI	-40 to 85	TM4C1232 C3PMI7
TM4C1232C3PMIR	Active	Production	LQFP (PM) | 64	1000 | LARGE T&R	Yes	NIPDAU	Level-3-260C-168 HR	-40 to 85	TM4C1232 C3PMI
TM4C1232C3PMIR.B	Active	Production	LQFP (PM) | 64	1000 | LARGE T&R	Yes	NIPDAU	Level-3-260C-168 HR	-40 to 85	TM4C1232 C3PMI

[1] **Status:** For more details on status, see our product life cycle.

[2] **Material type:** When designated, preproduction parts are prototypes/experimental devices, and are not yet approved or released for full production. Testing and final process, including without limitation quality assurance, reliability performance testing, and/or process qualification, may not yet be complete, and this item is subject to further changes or possible discontinuation. If available for ordering, purchases will be subject to an additional waiver at checkout, and are intended for early internal evaluation purposes only. These items are sold without warranties of any kind.

[3] **RoHS values:** Yes, No, RoHS Exempt. See the TI RoHS Statement for additional information and value definition.

[4] **Lead finish/Ball material:** Parts may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead finish/Ball material values may wrap to two lines if the finish value exceeds the maximum column width.

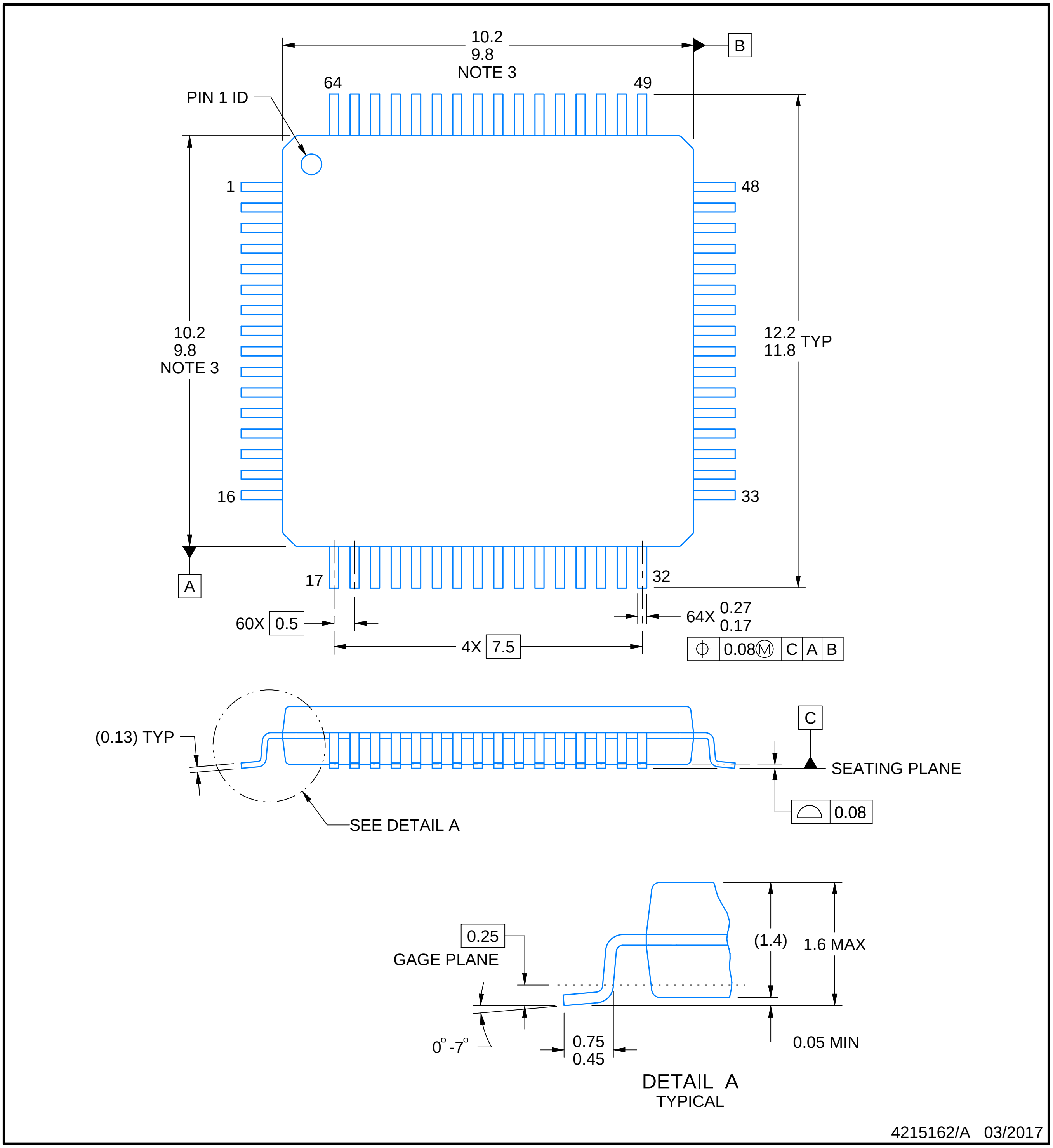
[5] **MSL rating/Peak reflow:** The moisture sensitivity level ratings and peak solder (reflow) temperatures. In the event that a part has multiple moisture sensitivity ratings, only the lowest level per JEDEC standards is shown. Refer to the shipping label for the actual reflow temperature that will be used to mount the part to the printed circuit board.

[6] **Part marking:** There may be an additional marking, which relates to the logo, the lot trace code information, or the environmental category of the part.

Multiple part markings will be inside parentheses. Only one part marking contained in parentheses and separated by a "~" will appear on a part. If a line is indented then it is a continuation of the previous line and the two combined represent the entire part marking for that device.

Important Information and Disclaimer: The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.



NOTES:

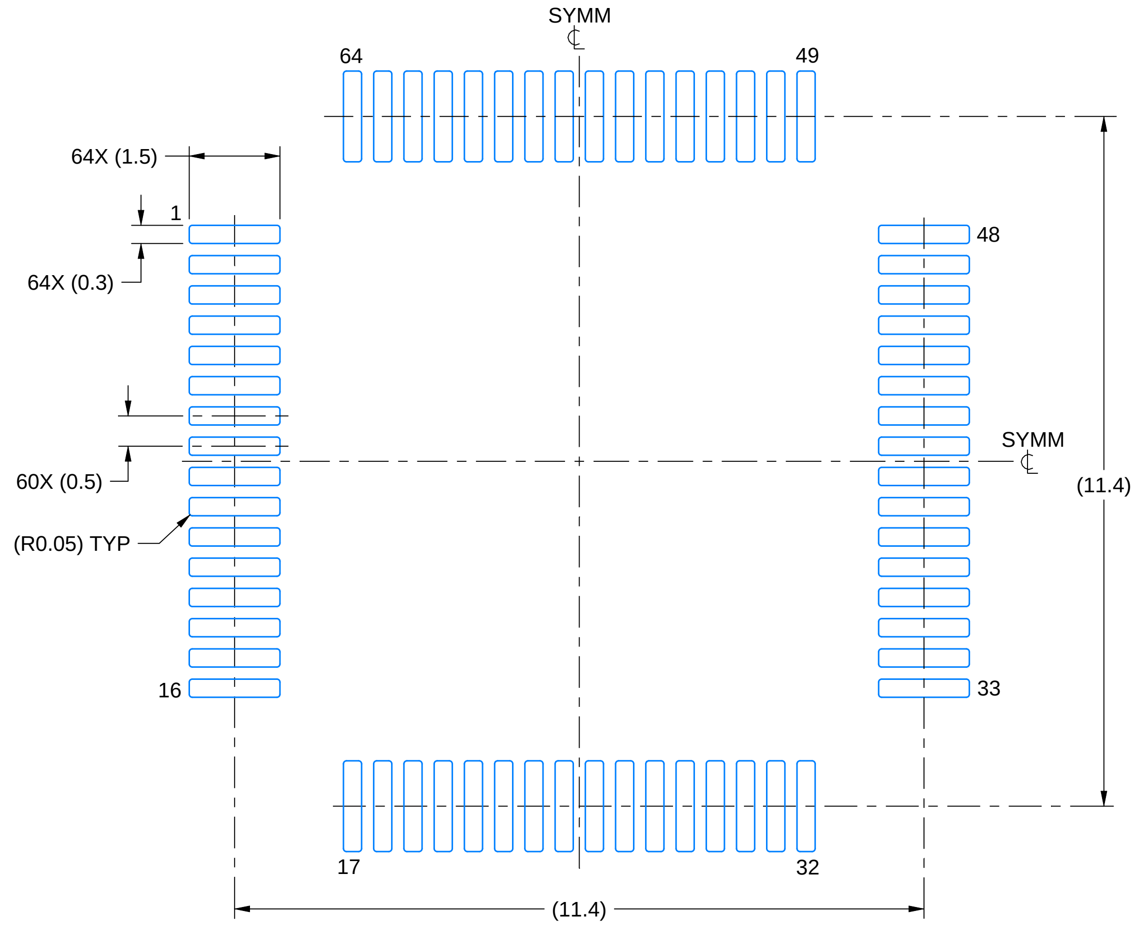
1. All linear dimensions are in millimeters. Any dimensions in parenthesis are for reference only. Dimensioning and tolerancing per ASME Y14.5M.
2. This drawing is subject to change without notice.
3. This dimension does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0.15 mm per side.
4. Reference JEDEC registration MS-026.

EXAMPLE BOARD LAYOUT

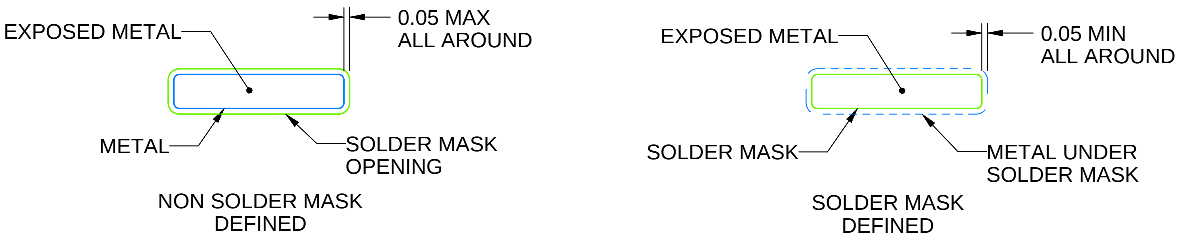
PM0064A

LQFP - 1.6 mm max height

PLASTIC QUAD FLATPACK



LAND PATTERN EXAMPLE
EXPOSED METAL SHOWN
SCALE:8X

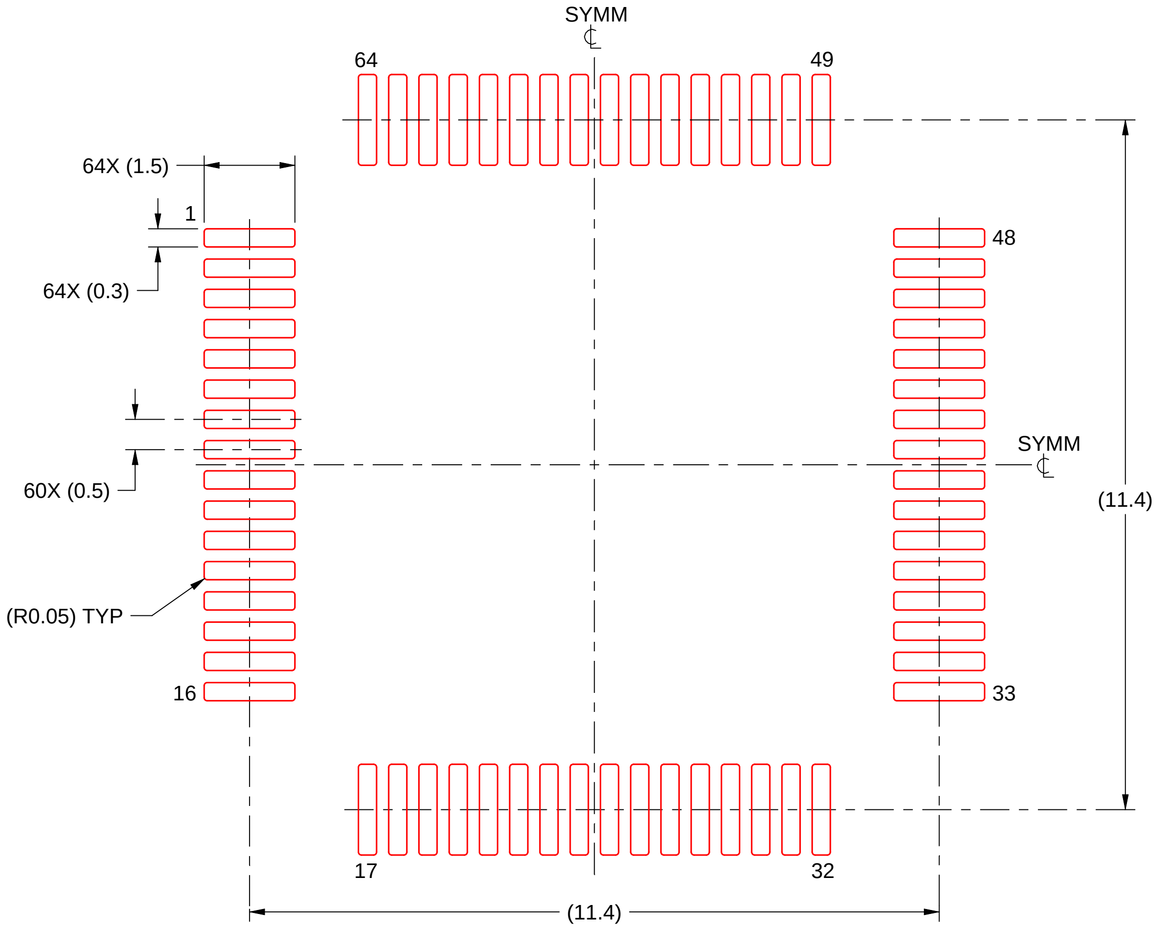


SOLDER MASK DETAILS

4215162/A 03/2017

NOTES: (continued)

5. Publication IPC-7351 may have alternate designs.
6. Solder mask tolerances between and around signal pads can vary based on board fabrication site.
7. For more information, see Texas Instruments literature number SLMA004 (www.ti.com/lit/slma004).



SOLDER PASTE EXAMPLE
BASED ON 0.125 mm THICK STENCIL
SCALE:8X

NOTES: (continued)

8. Laser cutting apertures with trapezoidal walls and rounded corners may offer better paste release. IPC-7525 may have alternate design recommendations.
9. Board assembly site may have different recommendations for stencil design.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025