Concerto F28M35x

Technical Reference Manual

Literature Number: SPRUH22I
April 2012–Revised November 2019

Contents

Preface **78**

1 System Control and Interrupts **79**

1.1 Signal Description 80
1.2 System Control Functional Description 81
1.2.1 Device Identification 81
1.2.2 Device Configuration Registers 82
1.3 Reset Control 82
1.3.1 Device Level Reset Sources 82
1.3.2 Handling of Resets at System Level 89
1.4 WIR Mode 92
1.4.1 Entering WIR Mode 93
1.4.2 Exiting WIR Mode 94
1.5 Exceptions and Interrupts Control 95
1.5.1 Master Subsystem Nested Vectored Interrupt Controller 95
1.5.2 Master Subsystem Exceptions Handling 96
1.5.3 Master Subsystem Non-Maskable Interrupt (MNMI) Module 97
1.5.4 Control Subsystem PIE 100
1.5.5 Control Subsystem Exceptions Handling 116
1.5.6 Control Subsystem NMI (CNMI) Module 116
1.6 Safety Features 119
1.6.1 Write Protection on Registers 119
1.6.2 Missing Clock Detection Logic 120
1.6.3 PLLSLIP Detection 123
1.6.4 Control Subsystem PIE Vector Address Validity Check 123
1.6.5 NMIWDs 124
1.6.6 Watchdog Timers 124
1.6.7 ECC and Parity Enabled RAMs, Shared RAMs Protection 124
1.6.8 ECC Enabled Flash Memory 125
1.7 Power Control 125
1.8 Clock Control 125
1.8.1 Clock Sources 126
1.8.2 PLLs 126
1.8.3 Master Subsystem Clocking 127
1.8.4 Control Subsystem Clocking 130
1.8.5 Clocking Control Semaphore Functionality 134
1.8.6 ACIB and Analog Peripherals Clocking 135
1.8.7 Configuring XCLKOUT 135
1.8.8 32-Bit CPU Timers 0/1/2 136
1.9 Low Power Modes 140
1.9.1 Low-Power Modes 140
1.10 Code Security Module (CSM) 144
1.10.1 Functional Description 144
1.10.2 CSM Impact on Other On-Chip Resources 147
1.10.3 Incorporating Code Security in User Applications 147
1.10.4 Do's and Don'ts to Protect Security Logic 155

Copyright © 2012–2019, Texas Instruments Incorporated

1.11 µCRC Module 155
1.11.1 Functional Description 155
1.11.2 CRC Polynomials 156
1.11.3 CRC Calculation Procedure 156
1.11.4 CRC Calculation For Data Stored In Secure Memory 156
1.12 Inter Processor Communications (IPC) 156
1.12.1 MSGRAMs 157
1.12.2 IPC Flags and Interrupts 157
1.12.3 MTOCIPC Communication 158
1.12.4 CTOMIPC Communication 159
1.12.5 Examples for Software IPC Procedure 160
1.12.6 IPC Message Registers 160
1.12.7 Flash Pump Semaphore 161
1.12.8 Clock Configuration Semaphore 162
1.12.9 Free Running Counter 164
1.13 System Control Registers 165
1.13.1 System Control, Configuration Register Map 165
1.13.2 Device Identification and Device Configuration 173
1.13.3 Reset Control and Status Registers 188
1.13.4 WIRMODE Registers 197
1.13.5 Exception and Interrupts 198
1.13.6 Safety Control Registers 218
1.13.7 Clocking Control Registers 222
1.13.8 Master Subsystem Code Security Module (CSM) Registers 248
1.13.9 Control Subsystem Code Security Module (CSM) Registers 264
1.13.10 µCRC Register Description 268
1.13.11 Master Subsystem IPC Registers 270
1.13.12 Control Subsystem IPC Registers 281
1.13.13 Master and Control Subsystem IPC Registers 293

2 M3 General-Purpose Timers 298

2.1 GPTM Features 299
2.2 Block Diagram 299
2.3 Functional Description 300
2.3.1 GPTM Reset Conditions 300
2.3.2 Timer Modes 301
2.3.3 DMA Operation 305
2.3.4 Accessing Concatenated Register Values 306
2.4 Initialization and Configuration 306
2.4.1 One-Shot/Periodic Timer Mode 306
2.4.2 Real-Time Clock (RTC) Mode 307
2.4.3 Input Edge-Count Mode 307
2.4.4 Input Edge Timing Mode 307
2.4.5 16-Bit PWM Mode 308
2.5 Register Map 308
2.6 Register Descriptions 309
2.6.1 GPTM Configuration (GPTMCFG) Register, offset 0x000 309
2.6.2 GPTM Timer A Mode (GPTMTAMR) Register, offset 0x004 309
2.6.3 GPTM Timer B Mode (GPTMTBMR) Register, offset 0x008 311
2.6.4 GPTM Control (GPTMCTL) Register, offset 0x00C 312
2.6.5 GPTM Interrupt Mask (GPTMIMR) Register, offset 0x018 313
2.6.6 GPTM Raw Interrupt Status (GPTMRIS) Register, offset 0x01C 314
2.6.7 GPTM Masked Interrupt Status (GPTMMIS) Register, offset 0x020 315
2.6.8 GPTM Interrupt Clear (GPTMICR) Register, offset 0x024 316

Copyright © 2012–2019, Texas Instruments Incorporated

2.6.9 GPTM Timer A Interval Load (GPTMTAILR) Register, offset 0x028 317
2.6.10 GPTM Timer B Interval Load (GPTMTBILR) Register, offset 0x02C 318
2.6.11 GPTM Timer A Match (GPTMTAMATCHR) Register, offset 0x030 318
2.6.12 GPTM Timer B Match (GPTMTBMATCHR) Register, offset 0x034 319
2.6.13 GPTM Timer A Prescale (GPTMTAPR) Register, offset 0x038 319
2.6.14 GPTM Timer B Prescale (GPTMTBPR) Register, offset 0x03C 320
2.6.15 GPTM Timer A Prescale Match (GPTMTAPMR) Register, offset 0x040 320
2.6.16 GPTM Timer B Prescale Match (GPTMTBPMR) Register, offset 0x044 320
2.6.17 GPTM Timer A (GPTMTAR) Register, offset 0x048 321
2.6.18 GPTM Timer B (GPTMTBR) Register, offset 0x04C 321
2.6.19 GPTM Timer A Value (GPTMTAV) Register, offset 0x050 322
2.6.20 GPTM Timer B Value (GPTMTBV) Register, offset 0x054 322

3 M3 Watchdog Timers 324

3.1 Introduction 325
3.1.1 Block Diagram 325
3.1.2 Functional Description 326
3.1.3 Initialization and Configuration 326
3.2 Register Map 326
3.3 Register Descriptions 327
3.3.1 Watchdog Load (WDTLOAD) Register, offset 0x000 327
3.3.2 Watchdog Value (WDTVALUE) Register, offset 0x004 328
3.3.3 Watchdog Control (WDTCTL) Register, offset 0x008 329
3.3.4 Watchdog Interrupt Clear (WDTICR) Register, offset 0x00C 329
3.3.5 Watchdog Raw Interrupt Status (WDTRIS) Register, offset 0x010 330
3.3.6 Watchdog Masked Interrupt Status (WDTMIS) Register, offset 0x014 330
3.3.7 Watchdog Test (WDTTEST) Register, offset 0x418 331
3.3.8 Watchdog Lock (WDTLOCK) Register, offset 0xC00 331
3.3.9 Watchdog Peripheral Identification 4 (WDTPeriphID4) Register, offset 0xFD0 332
3.3.10 Watchdog Peripheral Identification 5 (WDTPeriphID5) Register, offset 0xFD4 332
3.3.11 Watchdog Peripheral Identification 6 (WDTPeriphID6) Register, offset 0xFD8 332
3.3.12 Watchdog Peripheral Identification 7 (WDTPeriphID7) Register, offset 0xFDC 333
3.3.13 Watchdog Peripheral Identification 0 (WDTPeriphID0) Register, offset 0xFE0 333
3.3.14 Watchdog Peripheral Identification 1 (WDTPeriphID1) Register, offset 0xFE4 333
3.3.15 Watchdog Peripheral Identification 2 (WDTPeriphID2) Register, offset 0xFE8 334
3.3.16 Watchdog Peripheral Identification 3 (WDTPeriphID3) Register, offset 0xFEC 334
3.3.17 Watchdog PrimeCell Identification 0 (WDTPCellID0) Register, offset 0xFF0 334
3.3.18 Watchdog PrimeCell Identification 1 (WDTPCellID1) Register, offset 0xFF4 335
3.3.19 Watchdog PrimeCell Identification 2 (WDTPCellID2) Register, offset 0xFF8 335
3.3.20 Watchdog PrimeCell Identification 3 (WDTPCellID3) Register, offset 0xFFC 335

4 General-Purpose Input/Output (GPIO) 336

4.1 General-Purpose Input/Output (GPIO) 337
4.1.1 Introduction 337
4.1.2 Signal Description 337
4.1.3 Functional Description 341
4.1.4 Initialization and Configuration 345
4.1.5 Register Map 346
4.1.6 Register Descriptions 348
4.2 C28 General-Purpose Input/Output (GPIO) 369
4.2.1 Introduction 369
4.2.2 GPIO Module Overview 369
4.2.3 Configuration Overview 375
4.2.4 Digital General Purpose I/O Control 378
4.2.5 Input Qualification 380

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.6 GPIO and Peripheral Multiplexing (MUX) 384
4.2.7 Register Bit Definitions 391

5 Internal Memory 425
5.1 RAM Control Module 426
5.1.1 Functional Description 426
5.2 RAM Control Module Registers 434
5.2.1 M3 RAM Configuration Registers 437
5.2.2 M3 RAM Error Registers 452
5.2.3 C28x RAM Configuration Registers 465
5.2.4 C28x RAM Error Registers 480
5.3 Flash Controller Memory Module 493
5.3.1 Features 493
5.3.2 Flash Tools 493
5.3.3 Default Flash Configuration 494
5.3.4 Flash Bank, OTP and Pump 494
5.3.5 Flash Module Controller (FMC) 495
5.3.6 Flash and OTP Automatic Power-Down Modes 496
5.3.7 Flash and OTP Performance 497
5.3.8 Flash Read Interface 497
5.3.9 Erase/Program Flash 502
5.3.10 Error Correction Code (ECC) Protection 503
5.3.11 Reserved Locations Within Flash and OTP 506
5.3.12 Procedure to Change the Flash Control Registers 507
5.4 Flash Registers 508
5.4.1 Master Subsystem Flash Control Registers 512
5.4.2 Master Subsystem Flash ECC/Error Log Registers 518
5.4.3 Control Subsystem Flash Control Registers 525
5.4.4 Control Subsystem Flash ECC/Error Log Registers 529

6 ROM Code and Peripheral Booting 536
6.1 Introduction 537
6.2 Device Boot Sequence 537
6.3 Device Boot Modes 538
6.3.1 Configuring Alternate Boot Mode Pins 538
6.4 Device Boot Flow Diagram 540
6.5 M-Boot ROM Description 541
6.5.1 M-Boot ROM Memory Map 541
6.5.2 M-Boot ROM Vector Table 541
6.5.3 M-Boot ROM Version and Checksum Information 542
6.5.4 M-Boot ROM RAM Initialization 542
6.5.5 M-Boot ROM RAM Usage 543
6.5.6 M-Boot ROM User OTP 543
6.5.7 M-Boot ROM Entry Points 544
6.5.8 M-Boot ROM Clock Initialization 546
6.5.9 M-Boot ROM GPIO Assignments for Each Boot Mode 546
6.5.10 M-Boot ROM Functional Flow 547
6.5.11 M-Boot ROM Flow Diagram 549
6.5.12 M-Boot ROM Boot Status in RAM for Applications 552
6.5.13 M-Boot ROM Reset Cause Handling 553
6.5.14 M-Boot ROM Exceptions Handling 553
6.5.15 M-Boot ROM Boot Modes 554
6.6 C-Boot ROM Description 570
6.6.1 C-Boot ROM Memory Map 570
6.6.2 C-Boot ROM RAM Initialization 577

Copyright © 2012–2019, Texas Instruments Incorporated

6.6.3 C-Boot ROM RAM Usage 577
6.6.4 C-Boot ROM Boot Modes 578
6.6.5 C-Boot ROM Entry Points 579
6.6.6 C-Boot ROM GPIO Assignment for Boot Modes 580
6.6.7 C-Boot ROM Clock Initializations 581
6.6.8 C-Boot ROM Functional Flow 581
6.6.9 C-Boot ROM MTOC IPC Commands 584
6.6.10 C-Boot ROM Health Status 590
6.6.11 C-Boot ROM Boot Status 591
6.6.12 C-Boot ROM CTOM IPC Messages 592
6.6.13 C-Boot ROM Handling of Exceptions and PIE Interrupts 593
6.6.14 C-Boot ROM Bootloader Functionality 594
6.7 Guidelines for Boot ROM Application Writers 614
6.7.1 Master Subsystem Application Procedure to Start C-Boot ROM Bootloaders 614
6.7.2 Master Subsystem Application Procedure to Initialize the Control Subsystem RAM Using IPC 616
6.8 Application Notes to Use Boot ROM 619
6.8.1 How to Send Boot Data to M-Boot ROM 619
6.8.2 C-Boot ROM: Building the Boot Table 625
7 C28 Enhanced Pulse Width Modulator (ePWM) Module 630
7.1 Introduction 631
7.1.1 Submodule Overview 632
7.1.2 Register Mapping 636
7.2 ePWM Submodules 641
7.2.1 Overview 641
7.2.2 Time-Base (TB) Submodule 643
7.2.3 Counter-Compare (CC) Submodule 653
7.2.4 Action-Qualifier (AQ) Submodule 660
7.2.5 Dead-Band Generator (DB) Submodule 676
7.2.6 PWM-Chopper (PC) Submodule 683
7.2.7 Trip-Zone (TZ) Submodule 687
7.2.8 Event-Trigger (ET) Submodule 692
7.2.9 Digital Compare (DC) Submodule 699
7.3 Applications to Power Topologies 707
7.3.1 Overview of Multiple Modules 707
7.3.2 Key Configuration Capabilities 707
7.3.3 Controlling Multiple Buck Converters With Independent Frequencies 708
7.3.4 Controlling Multiple Buck Converters With Same Frequencies 712
7.3.5 Controlling Multiple Half H-Bridge (HHB) Converters 715
7.3.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM) 717
7.3.7 Practical Applications Using Phase Control Between PWM Modules 721
7.3.8 Controlling a 3-Phase Interleaved DC/DC Converter 722
7.3.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter 726
7.3.10 Controlling a Peak Current Mode Controlled Buck Module 728
7.3.11 Controlling H-Bridge LLC Resonant Converter 730
7.4 Registers 733
7.4.1 Time-Base Submodule Registers 733
7.4.2 Counter-Compare Submodule Registers 742
7.4.3 Action-Qualifier Submodule Registers 749
7.4.4 Dead-Band Submodule Registers 754
7.4.5 PWM-Chopper Submodule Control Register 758
7.4.6 Trip-Zone Submodule Control and Status Registers 760
7.4.7 Digital Compare Submodule Registers 767
7.4.8 GPTRIP Input Select Registers 778

Copyright © 2012–2019, Texas Instruments Incorporated

7.4.9 Event-Trigger Submodule Registers 780
7.4.10 Proper Interrupt Initialization Procedure 789
8 C28 Enhanced Capture (eCAP) Module 790
8.1 Introduction 791
8.2 Description 791
8.3 Capture and APWM Operating Mode 792
8.4 Capture Mode Description 793
8.4.1 Event Prescaler 794
8.4.2 Edge Polarity Select and Qualifier 795
8.4.3 Continuous/One-Shot Control 795
8.4.4 32-Bit Counter and Phase Control 796
8.4.5 CAP1-CAP4 Registers 797
8.4.6 Interrupt Control 797
8.4.7 Shadow Load and Lockout Control 798
8.4.8 APWM Mode Operation 799
8.5 Capture Module - Control and Status Registers 801
8.6 Register Mapping 809
8.7 Application of the ECAP Module 809
8.7.1 Example 1 - Absolute Time-Stamp Operation Rising Edge Trigger 810
8.7.2 Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger 812
8.7.3 Example 3 - Time Difference (Delta) Operation Rising Edge Trigger 814
8.7.4 Example 4 - Time Difference (Delta) Operation Rising and Falling Edge Trigger 816
8.8 Application of the APWM Mode 818
8.8.1 Example 1 - Simple PWM Generation (Independent Channel/s) 818
9 C28 Enhanced QEP (eQEP) Module 820
9.1 Introduction 821
9.2 Description 823
9.2.1 EQEP Inputs 823
9.2.2 Functional Description 824
9.2.3 eQEP Memory Map 824
9.3 Quadrature Decoder Unit (QDU) 826
9.3.1 Position Counter Input Modes 826
9.3.2 eQEP Input Polarity Selection 829
9.3.3 Position-Compare Sync Output 829
9.4 Position Counter and Control Unit (PCCU) 829
9.4.1 Position Counter Operating Modes 829
9.4.2 Position Counter Latch 831
9.4.3 Position Counter Initialization 833
9.4.4 eQEP Position-compare Unit 834
9.5 eQEP Edge Capture Unit 835
9.6 eQEP Watchdog 839
9.7 Unit Timer Base 839
9.8 eQEP Interrupt Structure 840
9.9 eQEP Registers 840
10 Analog Subsystem 854
10.1 Overview 855
10.2 Analog Common Interface Bus (ACIB) 855
10.3 Analog-to-Digital Converter (ADC) 859
10.3.1 Features 859
10.3.2 Block Diagram 860
10.3.3 SOC Principle of Operation 860
10.3.4 ONESHOT Single Conversion Support 864
10.3.5 ADC Conversion Priority 865

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.6 Simultaneous Sampling Mode 868
10.3.7 EOC and Interrupt Operation 868
10.3.8 Power Up Sequence 869
10.3.9 ADC Calibration 869
10.3.10 Internal/External Reference Voltage Selection 871
10.3.11 ADC Registers 872
10.3.12 Analog Subsystem Control Registers 889
10.3.13 ADC Timings 898
10.4 Comparator Block 901
10.4.1 Features 901
10.4.2 Comparator Block Diagram 902
10.4.3 Comparator Function 902
10.4.4 DAC Reference 902
10.4.5 Initialization 903
10.4.6 Digital Domain Manipulation 903
10.4.7 Comparator Registers 904
10.5 Analog Subsystem Software 906
10.5.1 C28 Analog Subsystem Functions 906
10.5.2 C28 Analog Subsystem Software Example 908
10.5.3 M3 Analog Subsystem Functions 908
10.5.4 M3 Analog Subsystem Software Example 909
11 C28 Direct Memory Access (DMA) Module 911
11.1 Introduction 912
11.2 Architecture 912
11.2.1 Block Diagram 912
11.2.2 Peripheral Interrupt Event Trigger Sources 913
11.2.3 DMA Bus 915
11.3 Pipeline Timing and Throughput 916
11.3.1 DMA Read Access of ADC Registers 917
11.4 CPU Arbitration 917
11.4.1 Arbitration when Accessing the Analog Subsystem 918
11.5 Channel Priority 918
11.5.1 Round-Robin Mode 918
11.5.2 Channel 1 High Priority Mode 919
11.6 Address Pointer and Transfer Control 920
11.7 Overrun Detection Feature 924
11.8 Register Descriptions 926
11.8.1 DMA Control Register (DMACTRL) — EALLOW Protected 927
11.8.2 Debug Control Register (DEBUGCTRL) — EALLOW Protected 929
11.8.3 Revision Register (REVISION) 929
11.8.4 Priority Control Register 1 (PRIORITYCTRL1) — EALLOW Protected 930
11.8.5 Priority Status Register (PRIORITYSTAT) 931
11.8.6 Mode Register (MODE) — EALLOW Protected 932
11.8.7 Control Register (CONTROL) — EALLOW Protected 934
11.8.8 Burst Size Register (BURST_SIZE) — EALLOW Protected 936
11.8.9 BURST_COUNT Register 936
11.8.10 Source Burst Step Register Size (SRC_BURST_STEP) — EALLOW Protected 937
11.8.11 Destination Burst Step Register Size (DST_BURST_STEP) — EALLOW Protected 938
11.8.12 Transfer Size Register (TRANSFER_SIZE) — EALLOW Protected 938
11.8.13 Transfer Count Register (TRANSFER_COUNT) 939
11.8.14 Source Transfer Step Size Register (SRC_TRANSFER_STEP) — EALLOW Protected 939
11.8.15 Destination Transfer Step Size Register (DST_TRANSFER_STEP) — EALLOW Protected 940
11.8.16 Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE) — EALLOW protected) 940

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.17 Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT) 941
11.8.18 Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP) — EALLOW Protected 941
11.8.19 Shadow Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW) — All EALLOW Protected 942
11.8.20 Active Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR/DST_BEG_ADDR) 942
11.8.21 Shadow Destination Begin and Current Address Pointer Registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW) — All EALLOW Protected 943
11.8.22 Active Destination Begin and Current Address Pointer Registers (SRC_ADDR/DST_ADDR) 943

12 C28 Serial Peripheral Interface (SPI) 944
12.1 Enhanced SPI Module Overview 945
12.1.1 SPI Block Diagram 946
12.1.2 SPI Module Signal Summary 948
12.1.3 Overview of SPI Module Registers 948
12.1.4 SPI Operation 949
12.1.5 SPI Interrupts 951
12.1.6 SPI FIFO Description 956
12.1.7 SPI 3-Wire Mode Description 958
12.1.8 SPI STEINV Bit in Digital Audio Transfers 960
12.2 C28 SPI-A to M3 SSI3 Internal Loopback 961
12.2.1 Loopback Initialization and Configuration 962
12.3 SPI Registers and Waveforms 964
12.3.1 SPI Control Registers 964
12.3.2 SPI Example Waveforms 973

13 C28 Serial Communications Interface (SCI) 979
13.1 Enhanced SCI Module Overview 980
13.1.1 Architecture 982
13.2 C28 SCI-A to M3 UART4 Internal Loopback 992
13.2.1 Loopback Initialization and Configuration 993
13.3 SCI Registers 994
13.3.1 SCI Module Register Summary 994
13.3.2 SCI Communication Control Register (SCICCR) 995
13.3.3 SCI Control Register 1 (SCICTL1) 996
13.3.4 SCI Baud-Select Registers (SCIHBAUD, SCILBAUD) 998
13.3.5 SCI Control Register 2 (SCICTL2) 999
13.3.6 SCI Receiver Status Register (SCIRXST) 999
13.3.7 Receiver Data Buffer Registers (SCIRXEMU, SCIRXBUF) 1001
13.3.8 SCI Transmit Data Buffer Register (SCITXBUF) 1002
13.3.9 SCI FIFO Registers (SCIFFTX, SCIFFRX, SCIFFCT) 1002
13.3.10 Priority Control Register (SCIPRI) 1006

14 C28 Inter-Integrated Circuit Module 1007
14.1 Introduction to the I2C Module 1008
14.1.1 Features 1009
14.1.2 Features Not Supported 1009
14.1.3 Functional Overview 1009
14.1.4 Clock Generation 1010
14.2 I2C Module Operational Details 1011
14.2.1 Input and Output Voltage Levels 1011
14.2.2 Data Validity 1011
14.2.3 Operating Modes 1012
14.2.4 I2C Module START and STOP Conditions 1013
14.2.5 Serial Data Formats 1013
14.2.6 NACK Bit Generation 1015

Copyright © 2012–2019, Texas Instruments Incorporated

14.2.7 Clock Synchronization 1016
14.2.8 Arbitration 1016
14.3 Interrupt Requests Generated by the I2C Module 1017
14.3.1 Basic I2C Interrupt Requests 1017
14.3.2 I2C FIFO Interrupts 1018
14.4 Resetting/Disabling the I2C Module 1018
14.5 I2C Module Registers 1019
14.5.1 I2C Mode Register (I2CMDR) 1020
14.5.2 I2C Extended Mode Register (I2CEMDR) 1023
14.5.3 I2C Interrupt Enable Register (I2CIER) 1025
14.5.4 I2C Status Register (I2CSTR) 1025
14.5.5 I2C Interrupt Source Register (I2CISRC) 1028
14.5.6 I2C Prescaler Register (I2CPSC) 1029
14.5.7 I2C Clock Divider Registers (I2CCLKL and I2CCLKH) 1029
14.5.8 I2C Slave Address Register (I2CSAR) 1031
14.5.9 I2C Own Address Register (I2COAR) 1031
14.5.10 I2C Data Count Register (I2CCNT) 1032
14.5.11 I2C Data Receive Register (I2CDRR) 1032
14.5.12 I2C Data Transmit Register (I2CDXR) 1032
14.5.13 I2C Transmit FIFO Register (I2CFFTX) 1033
14.5.14 I2C Receive FIFO Register (I2CFFRX) 1034
15 C28 Multichannel Buffered Serial Port (McBSP) 1036
15.1 Overview 1037
15.1.1 Features of the McBSP 1037
15.1.2 McBSP Pins/Signals 1038
15.1.3 McBSP Operation 1039
15.1.4 Data Transfer Process of McBSP 1040
15.1.5 Companding (Compressing and Expanding) Data 1040
15.2 Clocking and Framing Data 1042
15.2.1 Clocking 1042
15.2.2 Serial Words 1042
15.2.3 Frames and Frame Synchronization 1043
15.2.4 Generating Transmit and Receive Interrupts 1043
15.2.5 Ignoring Frame-Synchronization Pulses 1043
15.2.6 Frame Frequency 1044
15.2.7 Maximum Frame Frequency 1044
15.3 Frame Phases 1044
15.3.1 Number of Phases, Words, and Bits Per Frame 1045
15.3.2 Single-Phase Frame Example 1045
15.3.3 Dual-Phase Frame Example 1045
15.3.4 Implementing the AC97 Standard With a Dual-Phase Frame 1046
15.3.5 McBSP Reception 1046
15.3.6 McBSP Transmission 1048
15.3.7 Interrupts and DMA Events Generated by a McBSP 1049
15.4 McBSP Sample Rate Generator 1049
15.4.1 Block Diagram 1050
15.4.2 Frame Synchronization Generation in the Sample Rate Generator 1053
15.4.3 Synchronizing Sample Rate Generator Outputs to an External Clock 1053
15.4.4 Reset and Initialization Procedure for the Sample Rate Generator 1055
15.5 McBSP Exception/Error Conditions 1056
15.5.1 Types of Errors 1056
15.5.2 Overrun in the Receiver 1056
15.5.3 Unexpected Receive Frame-Synchronization Pulse 1058

Copyright © 2012–2019, Texas Instruments Incorporated

15.5.4 Overwrite in the Transmitter 1060
15.5.5 Unexpected Transmit Frame-Synchronization Pulse 1062
15.6 Multichannel Selection Modes 1064
15.6.1 Channels, Blocks, and Partitions 1064
15.6.2 Multichannel Selection 1065
15.6.3 Configuring a Frame for Multichannel Selection 1065
15.6.4 Using Two Partitions 1065
15.6.5 Using Eight Partitions 1067
15.6.6 Receive Multichannel Selection Mode 1068
15.6.7 Transmit Multichannel Selection Modes 1068
15.7 SPI Operation Using the Clock Stop Mode 1071
15.7.1 SPI Protocol 1071
15.7.2 Clock Stop Mode 1072
15.7.3 Bits Used to Enable and Configure the Clock Stop Mode 1072
15.7.4 Clock Stop Mode Timing Diagrams 1073
15.7.5 Procedure for Configuring a McBSP for SPI Operation 1075
15.7.6 McBSP as the SPI Master 1075
15.7.7 McBSP as an SPI Slave 1077
15.8 Receiver Configuration 1078
15.8.1 Programming the McBSP Registers for the Desired Receiver Operation 1078
15.8.2 Resetting and Enabling the Receiver 1079
15.8.3 Set the Receiver Pins to Operate as McBSP Pins 1079
15.8.4 Enable/Disable the Digital Loopback Mode 1080
15.8.5 Enable/Disable the Clock Stop Mode 1080
15.8.6 Enable/Disable the Receive Multichannel Selection Mode 1081
15.8.7 Choose One or Two Phases for the Receive Frame 1081
15.8.8 Set the Receive Word Length(s) 1082
15.8.9 Set the Receive Frame Length 1082
15.8.10 Enable/Disable the Receive Frame-Synchronization Ignore Function 1083
15.8.11 Set the Receive Companding Mode 1084
15.8.12 Set the Receive Data Delay 1085
15.8.13 Set the Receive Sign-Extension and Justification Mode 1087
15.8.14 Set the Receive Interrupt Mode 1088
15.8.15 Set the Receive Frame-Synchronization Mode 1088
15.8.16 Set the Receive Frame-Synchronization Polarity 1090
15.8.17 Set the Receive Clock Mode 1092
15.8.18 Set the Receive Clock Polarity 1093
15.8.19 Set the SRG Clock Divide-Down Value 1095
15.8.20 Set the SRG Clock Synchronization Mode 1095
15.8.21 Set the SRG Clock Mode (Choose an Input Clock) 1095
15.8.22 Set the SRG Input Clock Polarity 1097
15.9 Transmitter Configuration 1097
15.9.1 Programming the McBSP Registers for the Desired Transmitter Operation 1097
15.9.2 Resetting and Enabling the Transmitter 1098
15.9.3 Set the Transmitter Pins to Operate as McBSP Pins 1099
15.9.4 Enable/Disable the Digital Loopback Mode 1099
15.9.5 Enable/Disable the Clock Stop Mode 1099
15.9.6 Enable/Disable Transmit Multichannel Selection 1101
15.9.7 Choose One or Two Phases for the Transmit Frame 1101
15.9.8 Set the Transmit Word Length(s) 1101
15.9.9 Set the Transmit Frame Length 1103
15.9.10 Enable/Disable the Transmit Frame-Synchronization Ignore Function 1104
15.9.11 Set the Transmit Companding Mode 1105

Copyright © 2012–2019, Texas Instruments Incorporated

15.9.12 Set the Transmit Data Delay 1106
15.9.13 Set the Transmit DXENA Mode 1108
15.9.14 Set the Transmit Interrupt Mode 1108
15.9.15 Set the Transmit Frame-Synchronization Mode 1109
15.9.16 Set the Transmit Frame-Synchronization Polarity 1110
15.9.17 Set the SRG Frame-Synchronization Period and Pulse Width 1111
15.9.18 Set the Transmit Clock Mode 1112
15.9.19 Set the Transmit Clock Polarity 1112
15.10 Emulation and Reset Considerations 1114
15.10.1 McBSP Emulation Mode 1114
15.10.2 Resetting and Initializing McBSP 1114
15.11 Data Packing Examples 1116
15.11.1 Data Packing Using Frame Length and Word Length 1116
15.11.2 Data Packing Using Word Length and the Frame-Synchronization Ignore Function 1118
15.12 McBSP Registers 1118
15.12.1 Register Summary 1119
15.12.2 Data Receive Registers (DRR[1,2]) 1119
15.12.3 Data Transmit Registers (DXR[1,2]) 1120
15.12.4 Serial Port Control Registers (SPCR[1,2]) 1121
15.12.5 Receive Control Registers (RCR[1, 2]) 1126
15.12.6 Transmit Control Registers (XCR1 and XCR2) 1128
15.12.7 Sample Rate Generator Registers (SRGR1 and SRGR2) 1131
15.12.8 Multichannel Control Registers (MCR[1,2]) 1133
15.12.9 Pin Control Register (PCR) 1138
15.12.10 Receive Channel Enable Registers (RCERA, RCERB, RCERC, RCERD, RCERE, RCERF, RCERG, RCERH) 1140
15.12.11 Transmit Channel Enable Registers (XCERA, XCERB, XCERC, XCERD, XCERE, XCERF, XCERG, XCERH) 1142
15.12.12 Interrupt Generation 1144
16 M3 Micro Direct Memory Access (µDMA) 1148
16.1 Overview 1149
16.2 Block Diagram 1149
16.3 Functional Description 1150
16.3.1 Channel Assignments 1151
16.3.2 Priority 1152
16.3.3 Arbitration Size 1152
16.3.4 Request Types 1152
16.3.5 Channel Configuration 1153
16.3.6 Transfer Modes 1154
16.3.7 Transfer Size and Increment 1163
16.3.8 Peripheral Interface 1163
16.3.9 Software Request 1163
16.3.10 Interrupts and Errors 1164
16.4 Initialization and Configuration 1164
16.4.1 Module Initialization 1164
16.4.2 Configuring a Memory-to-Memory Transfer 1164
16.4.3 Configuring a Peripheral for Simple Transmit 1165
16.4.4 Configuring a Peripheral for Ping-Pong Receive 1167
16.5 Register Map 1169
16.6 µDMA Channel Control Structure 1171
16.6.1 DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000 1171
16.6.2 DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004 1171
16.6.3 DMA Channel Control Word (DMACHCTL), offset 0x008 1172
16.7 µDMA Register Descriptions 1175

Copyright © 2012–2019, Texas Instruments Incorporated

16.7.1 DMA Status (DMASTAT), offset 0x000 1175
16.7.2 DMA Configuration (DMACFG), offset 0x004 1176
16.7.3 DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008 1176
16.7.4 DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C 1177
16.7.5 DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010 1177
16.7.6 DMA Channel Software Request (DMASWREQ), offset 0x014 1177
16.7.7 DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018 1178
16.7.8 DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C 1178
16.7.9 DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020 1179
16.7.10 DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024 1179
16.7.11 DMA Channel Enable Set (DMAENASET), offset 0x028 1179
16.7.12 DMA Channel Enable Clear (DMAENACLR), offset 0x02C 1180
16.7.13 DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030 1180
16.7.14 DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034 1181
16.7.15 DMA Channel Priority Set (DMAPRIOSET), offset 0x038 1181
16.7.16 DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C 1181
16.7.17 DMA Bus Error Clear (DMAERRCLR), offset 0x04C 1182
16.7.18 DMA Channel Assignment (DMACHALT), offset 0x500 1182
16.7.19 DMA Channel Map Assignment (DMACHMAP0) Register, offset 0x510 1182
16.7.20 DMA Channel Map Assignment (DMACHMAP1) Register, offset 0x514 1183
16.7.21 DMA Channel Map Assignment (DMACHMAP2) Register, offset 0x518 1184
16.7.22 DMA Channel Map Assignment (DMACHMAP3) Register, offset 0x51C 1185
16.7.23 DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0 1186
16.7.24 DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4 1187
16.7.25 DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8 1187
16.7.26 DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC 1187
16.7.27 DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0 1188
16.7.28 DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0 1188
16.7.29 DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4 1188
16.7.30 DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8 1189
16.7.31 DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC 1189

17 External Peripheral Interface (EPI) 1190

17.1 Introduction 1191
17.2 EPI Block Diagram 1192
17.3 Functional Description 1192
17.3.1 Non-Blocking Reads 1193
17.4 DMA Operation 1194
17.5 Initialization and Configuration 1194
17.6 SDRAM Mode 1195
17.6.1 External Signal Connections 1196
17.6.2 Refresh Configuration 1196
17.6.3 Bus Interface Speed 1197
17.6.4 Non-Blocking Read Cycle 1197
17.6.5 Normal Read Cycle 1197
17.6.6 Write Cycle 1198
17.7 Host Bus Mode 1199
17.7.1 Control Pins 1199
17.7.2 Speed of Transactions 1210
17.7.3 Sub-Modes of Host Bus 8/16 1211
17.7.4 Host Bus Operation 1212
17.8 General-Purpose Mode 1215
17.8.1 General Purpose Bus Operation 1218
17.9 C28x Access to EPI 1223

Copyright © 2012–2019, Texas Instruments Incorporated

17.9.1 Real-Time Window (RTW) 1224
17.9.2 Memory Protection 1224
17.10 Register Map 1225
17.11 Register Descriptions 1228
17.11.1 EPI Configuration Register (EPICFG), offset 0x000 1228
17.11.2 EPI Main Baud Rate (EPIBAUD) Register, offset 0x004 1229
17.11.3 EPI Main Baud Rate (EPIBAUD2) Register, offset 0x008 1230
17.11.4 EPI SDRAM Configuration (EPISDRAMCFG) Register, offset 0x010 1231
17.11.5 EPI Host-Bus 8 Configuration (EPIHB8CFG) Register, offset 0x010 1232
17.11.6 EPI Host-Bus 16 Configuration (EPIHB16CFG) Register. offset 0x010 1235
17.11.7 EPI General-Purpose Configuration (EPIGPCFG) Register, offset 0x010 1238
17.11.8 EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2) Register, Offset 0x014 1241
17.11.9 EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2) Register, offset 0x014 1244
17.11.10 EPI General-Purpose Configuration 2 (EPIGPCFG2) Register, offset 0x014 1247
17.11.11 EPI Address Map (EPIADDRMAP) Register, offset 0x01C 1248
17.11.12 EPI Read Size 0 (EPIRSIZE0) Register and EPI Read Size 1 (EPIRSIZE1) Register, offset 0x020 and 0x030 1249
17.11.13 EPI Read Address 0 (EPIRADDR0) Register and EPI Read Address 1 (EPIRADDR1) Register, offset 0x024 and 0x034 1251
17.11.14 EPI Non-Blocking Read Data 0 (EPIRPSTD0) Register and EPI Non-Blocking Read Data 1 (EPIRPSTD1) Register, offset 0x028 and 0x038 1251
17.11.15 EPI Status (EPISTAT) Register, offset 0x060 1253
17.11.16 EPI Read FIFO Count (EPIRFIFOCNT) Register, offset 0x06C 1254
17.11.17 EPI Read FIFO (EPIREADFIFO) Register and EPI Read FIFO Alias 1-7 (EPIREADFIFO1-7) Registers, offset 0x070 and 0x08C 1255
17.11.18 EPI FIFO Level Selects (EPIFIFOLVL) Register, 0x200 1255
17.11.19 EPI Write FIFO Count (EPIWFIFOCNT) Register, offset 0x204 1257
17.11.20 EPI DMA Transmit Count (EPIDMATXCNT) Register, offset 0x208 1258
17.11.21 EPI Interrupt Mask (EPIIM) Register, 0x210 1258
17.11.22 EPI Raw Interrupt Status (EPIRIS) Register, offset 0x214 1259
17.11.23 EPI Masked Interrupt Status (EPIMIS) Register, offset 0x218 1260
17.11.24 EPI Error Interrupt Status and Clear (EPIEISC) Register, offset 0x21C 1262
17.11.25 EPI Host-Bus 8 Configuration 3 (EPIHB8CFG3), offset 0x308 1263
17.11.26 EPI Host-Bus 16 Configuration 3 (EPIHB16CFG3), offset 0x308 1264
17.11.27 EPI Host-Bus 8 Configuration 4 (EPIHB8CFG4), offset 0x30C 1265
17.11.28 EPI Host-Bus 16 Configuration 4 (EPIHB16CFG4), offset 0x30C 1267
17.11.29 EPI Host-Bus 8 Timing Extension (EPIHB8TIME), offset 0x310 1268
17.11.30 EPI Host-Bus 16 Timing Extension (EPIHB16TIME), offset 0x310 1269
17.11.31 EPI Host-Bus 8 Timing Extension (EPIHB8TIME2), offset 0x314 1270
17.11.32 EPI Host-Bus 16 Timing Extension (EPIHB16TIME2), offset 0x314 1271
17.11.33 EPI Host-Bus 8 Timing Extension (EPIHB8TIME3), offset 0x318 1272
17.11.34 EPI Host-Bus 16 Timing Extension (EPIHB16TIME3) Register, offset 0x318 1273
17.11.35 EPI Host-Bus 8 Timing Extension (EPIHB8TIME4) Register, offset 0x31C 1274
17.11.36 EPI Host-Bus 16 Timing Extension (EPIHB16TIME4) Register, 0x31C 1275
17.11.37 CEPIRTWCFG Register 1276
17.11.38 CEPIRTWCNT Register 1276
17.11.39 CEPIRTWPRD Register 1277
17.11.40 CEPISTATUS Register 1277
17.11.41 MEMPROT Register 1277
18 M3 Universal Serial Bus (USB) Controller 1279
18.1 Introduction 1280
18.1.1 Block Diagram 1280
18.2 Functional Description 1280
18.2.1 Operation as a Device 1281

Copyright © 2012–2019, Texas Instruments Incorporated

18.2.2 Operation as a Host 1285
18.2.3 OTG Mode 1288
18.2.4 DMA Operation 1290
18.3 Initialization and Configuration 1291
18.3.1 Pin Configuration 1291
18.3.2 Endpoint Configuration 1291
18.4 Register Map 1292
18.5 Register Descriptions 1300
18.5.1 USB Device Functional Address Register (USBFADDR), offset 0x000 1300
18.5.2 USB Power Management Register (USBPOWER), offset 0x001 1301
18.5.3 USB Transmit Interrupt Status Register (USBTXIS), offset 0x002 1303
18.5.4 USB Receive Interrupt Status Register (USBRXIS), offset 0x004 1305
18.5.5 USB Transmit Interrupt Enable Register (USBTXIE), offset 0x006 1307
18.5.6 USB Receive Interrupt Enable Register (USBRXIE), offset 0x008 1309
18.5.7 USB General Interrupt Status Register (USBIS), offset 0x00A 1311
18.5.8 USB Interrupt Enable Register (USBIE), offset 0x00B 1313
18.5.9 USB Frame Value Register (USBFRAME), offset 0x00C 1315
18.5.10 USB Endpoint Index Register (USBEPIDX), offset 0x00E 1315
18.5.11 USB Test Mode Register (USBTEST), offset 0x00F 1316
18.5.12 USB FIFO Endpoint n Register (USBFIFO[0]-USBFIFO[15]) 1318
18.5.13 USB Device Control Register (USBDEVCTL), offset 0x060 1319
18.5.14 USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ), offset 0x062 1321
18.5.15 USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ), offset 0x063 1322
18.5.16 USB Transmit FIFO Start Address Register (USBTXFIFOADD), offset 0x064 1323
18.5.17 USB Receive FIFO Start Address Register (USBRXFIFOADD), offset 0x066 1324
18.5.18 USB Connect Timing Register (USBCONTIM), offset 0x07A 1325
18.5.19 USB OTG VBUS Pulse Timing Register (USBVPLEN), offset 0x07B 1325
18.5.20 USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF), offset 0x07D 1326
18.5.21 USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF), offset 0x07E 1326
18.5.22 USB Transmit Functional Address Endpoint n Registers (USBTXFUNCADDR[0]-USBTXFUNCADDR[15]) 1327
18.5.23 USB Transmit Hub Address Endpoint n Registers (USBTXHUBADDR[0]-USBTXHUBADDR[15]) 1328
18.5.24 USB Transmit Hub Port Endpoint n Registers (USBTXHUBPORT[0]-USBTXHUBPORT[15]) 1329
18.5.25 USB Receive Functional Address Endpoint n Registers (USBRXFUNCADDR[1]-USBRXFUNCADDR[15]) 1330
18.5.26 USB Receive Hub Address Endpoint n Registers (USBRXHUBADDR[1]-USBRXHUBADDR[15]) 1331
18.5.27 USB Receive Hub Port Endpoint n Registers (USBRXHUBPORT[1]-USBRXHUBPORT[15]) 1332
18.5.28 USB Maximum Transmit Data Endpoint n Registers (USBTXMAXP[1]-USBTXMAXP[15]) 1333
18.5.29 USB Control and Status Endpoint 0 Low Register (USBCSRL0), offset 0x102 1334
18.5.30 USB Control and Status Endpoint 0 High Register (USBCSRH0), offset 0x103 1336
18.5.31 USB Receive Byte Count Endpoint 0 Register (USBCOUNT0), offset 0x108 1337
18.5.32 USB Type Endpoint 0 Register (USBTYPE0), offset 0x10A 1337
18.5.33 USB NAK Limit Register (USBNAKLMT), offset 0x10B 1338
18.5.34 USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[1]-USBTXCSRL[15]) 1339
18.5.35 USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[1]-USBTXCSRH[15]) 1342
18.5.36 USB Maximum Receive Data Endpoint n Registers (USBRXMAXP[1]-USBRXMAXP[15]) 1344
18.5.37 USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[1]-USBRXCSRL[15]) 1345
18.5.38 USB Receive Control and Status Endpoint n High Register (USBRXCSRH[1]-

Copyright © 2012–2019, Texas Instruments Incorporated

USBRXCSRH[15]) 1348
18.5.39 USB Receive Byte Count Endpoint n Registers (USBRXCOUNT[1]-USBRXCOUNT[15]) 1350
18.5.40 USB Host Transmit Configure Type Endpoint n Register (USBTXTYPE[1]-USBTXTYPE[15]) ... 1351
18.5.41 USB Host Transmit Interval Endpoint n Register (USBTXINTERVAL[1]USBTXINTERVAL[15]) . 1352
18.5.42 USB Host Configure Receive Type Endpoint n Register (USBRXTYPE[1]-USBRXTYPE[15]) ... 1353
18.5.43 USB Host Receive Polling Interval Endpoint n Register (USBRXINTERVAL[1]USBRXINTERVAL[15]) 1354
18.5.44 USB Request Packet Count in Block Transfer Endpoint n Registers (USBRQPKTCOUNT[1]USBRQPKTCOUNT[15]) 1355
18.5.45 USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS), offset 0x340 1356
18.5.46 USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS), offset 0x342 1358
18.5.47 USB External Power Control Register (USBEPC), offset 0x400 1360
18.5.48 USB External Power Control Raw Interrupt Status Register (USBEPCRIS), offset 0x404 1362
18.5.49 USB External Power Control Interrupt Mask Register (USBEPCIM), offset 0x408 1363
18.5.50 USB External Power Control Interrupt Status and Clear Register (USBEPCISC), offset 0x40C . 1364
18.5.51 USB Device RESUME Raw Interrupt Status Register (USBDRRIS), offset 0x410 1365
18.5.52 USB Device RESUME Raw Interrupt Mask Register (USBDRIM), offset 0x414 1366
18.5.53 USB Device RESUME Interrupt Status and Clear Register (USBDRISC), offset 0x418 1367
18.5.54 USB General-Purpose Control and Status Register (USBGPCS), offset 0x41C 1368
18.5.55 USB VBUS Droop Control Register (USBVDC), offset 0x430 1369
18.5.56 USB VBUS Droop Control Raw Interrupt Status Register (USBVDCRIS), offset 0x434 1370
18.5.57 USB VBUS Droop Control Interrupt Mask Register (USBVDCIM), offset 0x438 1371
18.5.58 USB VBUS Droop Control Interrupt Status and Clear Register (USBVDCISC), offset 0x43C 1372
18.5.59 USB ID Valid Detect Raw Interrupt Status Register (USBIDVRIS), offset 0x444 1373
18.5.60 USB ID Valid Detect Interrupt Mask Register (USBIDVIM), offset 0x448 1374
18.5.61 USB ID Valid Detect Interrupt Status and Clear Register (USBIDVISC), offset 0x44C 1375
18.5.62 USB DMA Select Register (USBDMASEL), offset 0x450 1376

19 M3 Ethernet Media Access Controller (EMAC) 1380

19.1 Introduction 1381
19.2 EMAC Block Diagram 1381
19.3 Functional Description 1382
19.3.1 MAC Operation 1382
19.3.2 Internal MII Operation 1385
19.3.3 Interrupts 1385
19.3.4 DMA Operation 1386
19.4 Initialization and Configuration 1386
19.4.1 Software Configuration 1386
19.5 Register Map 1387
19.6 Ethernet MAC Register Descriptions 1388
19.6.1 Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK) Register, offset 0x000 .. 1388
19.6.2 Ethernet MAC Interrupt Mask (MACIM) Register, offset 0x004 1389
19.6.3 Ethernet MAC Receive Control (MACRCTL) Register, offset 0x008 1390
19.6.4 Ethernet MAC Transmit Control (MACTCTL) Register, offset 0x00C 1391
19.6.5 Ethernet MAC Data (MACDATA) Register, offset 0x010 1393
19.6.6 Ethernet MAC Individual Address 0 (MACIA0) Register, offset 0x014 1394
19.6.7 Ethernet MAC Individual Address 1 (MACIA1) Register, offset 0x018 1395
19.6.8 Ethernet MAC Threshold (MACTHR) Register, offset 0x01C 1396
19.6.9 Ethernet MAC Management Control (MACMCTL) Register, offset 0x020 1397
19.6.10 Ethernet MAC Management Divider (MACMDV) Register, offset 0x024 1398
19.6.11 Ethernet MAC Management Address Register (MACMAR), offset 0x028 1398
19.6.12 Ethernet MAC Management Transmit Data (MACMTXD) Register, offset 0x02C 1398
19.6.13 Ethernet MAC Management Receive Data (MACMRXD) Register, offset 0x030 1400
19.6.14 Ethernet MAC Number of Packets (MACNP) Register, offset 0x034 1400
19.6.15 Ethernet MAC Transmission Request (MACTR) Register, offset 0x038 1401

Copyright © 2012–2019, Texas Instruments Incorporated

19.6.16 Ethernet MAC Timer Support (MACTS) Register, offset 0x03C ... 1401
19.7 MII Management Register Descriptions ... 1402
19.7.1 Ethernet PHY Management Register 0 – Control (MR0) Register, address 0x00 ... 1402
19.7.2 Ethernet PHY Management Register 1 – Status (MR1) Register, address 0x01 ... 1404
19.7.3 Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2) Register , address 0x02 ... 1405
19.7.4 Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3) Register, address 0x03 ... 1405
19.7.5 Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4) Register, address 0x04 ... 1406
19.7.6 Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5) Register, address 0x05 ... 1407
19.7.7 Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6) Register, address 0x06 ... 1408

20 M3 Synchronous Serial Interface (SSI) ... 1409
20.1 Introduction ... 1410
20.2 Features ... 1410
20.2.1 SSI Block Diagram ... 1410
20.3 Functional Description ... 1411
20.3.1 Bit Rate Generation ... 1412
20.3.2 FIFO Operation ... 1412
20.3.3 Interrupts ... 1412
20.3.4 Frame Formats ... 1413
20.3.5 DMA Operation ... 1417
20.4 Initialization and Configuration ... 1418
20.5 SSI Registers ... 1419
20.5.1 SSI Base Addresses ... 1419
20.5.2 SSI_REGS Registers ... 1420

21 M3 Universal Asynchronous Receivers/Transmitters (UARTs) ... 1451
21.1 Introduction ... 1452
21.2 Block Diagram ... 1452
21.3 Functional Description ... 1453
21.3.1 Transmit/Receive Logic ... 1453
21.3.2 Baud-Rate Generation ... 1454
21.3.3 Data Transmission ... 1454
21.3.4 Serial IR (SIR) ... 1455
21.3.5 ISO 7816 Support ... 1456
21.3.6 LIN Support ... 1456
21.3.7 FIFO Operation ... 1457
21.3.8 Interrupts ... 1457
21.3.9 Loopback Operation ... 1458
21.3.10 DMA Operation ... 1458
21.4 M3 UART4 to C28 SCI-A Internal Loopback ... 1458
21.4.1 Loopback Initialization and Configuration ... 1459
21.5 Initialization and Configuration ... 1459
21.6 Register Map ... 1460
21.7 Register Descriptions ... 1461
21.7.1 UART Data Register (UARTDR), offset 0x000 ... 1461
21.7.2 UART Receive Status/Error Clear Register (UARTRSR/UARTECR), offset 0x004 ... 1462
21.7.3 UART Flag Register (UARTFR), offset 0x018 ... 1464
21.7.4 UART IrDA Low-Power Register (UARTILPR), offset 0x020 ... 1466
21.7.5 UART Integer Baud-Rate Divisor Register (UARTIBRD), offset 0x024 ... 1466
21.7.6 UART Fractional Baud-Rate Divisor Register (UARTFBRD), offset 0x028 ... 1467
21.7.7 UART Line Control Register (UARTLCRH), offset 0x02C ... 1467
21.7.8 UART Control Register (UARTCTL), offset 0x030 ... 1468

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.9 UART Interrupt FIFO Level Select (UARTIFLS) Register, offset 0x034 1470
21.7.10 UART Interrupt Mask (UARTIM) Register, offset 0x038 1471
21.7.11 UART Raw Interrupt Status (UARTRIS), offset 0x03C 1473
21.7.12 UART Masked Interrupt Status (UARTMIS), offset 0x040 1475
21.7.13 UART Interrupt Clear (UARTICR), offset 0x044 1477
21.7.14 UART DMA Control (UARTDMACTL), offset 0x048 1478
21.7.15 UART LIN Control (UARTLCTL), offset 0x090 1478
21.7.16 UART LIN Snap Shot (UARTLSS), offset 0x094 1479
21.7.17 UART LIN Timer (UARTLTIM), offset 0x098 1479
21.7.18 UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0 1480
21.7.19 UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4 1480
21.7.20 UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8 1480
21.7.21 UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC 1481
21.7.22 UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0 1481
21.7.23 UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4 1481
21.7.24 UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8 1482
21.7.25 UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC 1482
21.7.26 UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0 1482
21.7.27 UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4 1483
21.7.28 UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8 1483
21.7.29 UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC 1483

22 M3 Inter-Integrated Circuit (I2C) Interface 1484

22.1 Introduction 1485
22.2 I2C Block Diagram 1485
22.3 Functional Description 1485
22.3.1 I2C Bus Functional Overview 1486
22.3.2 Available Speed Modes 1487
22.3.3 Interrupts 1489
22.3.4 Loopback Operation 1490
22.3.5 Command Sequence Flow Charts 1490
22.4 Initialization and Configuration 1497
22.5 Register Map 1498
22.6 Register Descriptions 1499
22.6.1 I2C Master Slave Address (I2CMSA), offset 0x000 1499
22.6.2 I2C Master Control/Status (I2CMCS), offset 0x004 1500
22.6.3 I2C Master Data (I2CMDR), offset 0x008 1504
22.6.4 I2C Master Timer Period (I2CMTPR), offset 0x00C 1504
22.6.5 I2C Master Interrupt Mask (I2CMIMR), offset 0x010 1505
22.6.6 I2C Master Raw Interrupt Status (I2CMRIS), offset 0x014 1505
22.6.7 I2C Master Masked Interrupt Status (I2CMMIS), offset 0x018 1506
22.6.8 I2C Master Interrupt Clear (I2CMICR), offset 0x01C 1506
22.6.9 I2C Master Configuration (I2CMCR), offset 0x020 1507
22.7 Register Descriptions (I2C Slave) 1508
22.7.1 I2C Slave Own Address (I2CSOAR), offset 0x800 1508
22.7.2 I2C Slave Control/Status (I2CSCSR), offset 0x804 1508
22.7.3 I2C Slave Data (I2CSDR), offset 0x808 1509
22.7.4 I2C Slave Interrupt Mask (I2CSIMR), offset 0x80C 1509
22.7.5 I2C Slave Raw Interrupt Status (I2CSRIS), offset 0x810 1510
22.7.6 I2C Slave Masked Interrupt Status (I2CSMIS), offset 0x814 1510
22.7.7 I2C Slave Interrupt Clear (I2CSICR), offset 0x818 1511

23 M3 Controller Area Network (CAN) 1512

23.1 Overview 1513
23.1.1 Features 1513

Copyright © 2012–2019, Texas Instruments Incorporated

23.1.2 Functional Description 1513
23.1.3 Block Diagram 1514
23.2 Operating Modes 1515
23.2.1 Initialization 1515
23.2.2 CAN Message Transfer (Normal Operation) 1515
23.2.3 Test Modes 1516
23.3 Multiple Clock Source 1520
23.4 Interrupt Functionality 1520
23.4.1 Message Object Interrupts 1520
23.4.2 Status Change Interrupts 1520
23.4.3 Error Interrupts 1521
23.5 Global Power-down Mode 1521
23.5.1 Entering Global Power-down Mode 1521
23.5.2 Wakeup from Global Power-down Mode 1521
23.6 Local Power-down Mode 1521
23.6.1 Entering Local Power-down Mode 1521
23.6.2 Wakeup from Local Power-down Mode 1521
23.7 Parity Check Mechanism 1522
23.7.1 Behavior on Parity Error 1522
23.8 Debug Mode 1522
23.9 Module Initialization 1523
23.10 Configuration of Message Objects 1523
23.10.1 Configuration of a Transmit Object for Data Frames 1523
23.10.2 Configuration of a Transmit Object for Remote Frames 1524
23.10.3 Configuration of a Single Receive Object for Data Frames 1524
23.10.4 Configuration of a Single Receive Object for Remote Frames 1524
23.10.5 Configuration of a FIFO Buffer 1525
23.11 Message Handling 1525
23.11.1 Message Handler Overview 1525
23.11.2 Receive/Transmit Priority 1526
23.11.3 Transmission of Messages in Event Driven CAN Communication 1526
23.11.4 Updating a Transmit Object 1526
23.11.5 Changing a Transmit Object 1527
23.11.6 Acceptance Filtering of Received Messages 1527
23.11.7 Reception of Data Frames 1527
23.11.8 Reception of Remote Frames 1527
23.11.9 Reading Received Messages 1528
23.11.10 Requesting New Data for a Receive Object 1528
23.11.11 Storing Received Messages in FIFO Buffers 1528
23.11.12 Reading from a FIFO Buffer 1528
23.12 CAN Bit Timing 1529
23.12.1 Bit Time and Bit Rate 1530
23.12.2 Configuration of the CAN Bit Timing 1535
23.13 Message Interface Register Sets 1537
23.13.1 Message Interface Register Sets 1 and 2 1538
23.13.2 IF3 Register Set 1539
23.14 Message RAM 1539
23.14.1 Structure of Message Objects 1540
23.14.2 Addressing Message Objects in RAM 1542
23.14.3 Message RAM Representation in Debug Mode 1543
23.15 CAN Control Registers 1543
23.15.1 CAN Control Register (CAN CTL) 1544
23.15.2 Error and Status Register (CAN ES) 1546

Copyright © 2012–2019, Texas Instruments Incorporated

23.15.3 Error Counter Register (CAN ERRC) 1548
23.15.4 Bit Timing Register (CAN BTR) 1548
23.15.5 Interrupt Register (CAN INT) 1549
23.15.6 Test Register (CAN TEST) 1550
23.15.7 Parity Error Code Register (CAN PERR) 1551
23.15.8 Auto-Bus-On Time Register (CAN ABOTR) 1551
23.15.9 Transmission Request Registers (CAN TXRQ) 1552
23.15.10 New Data Registers (CAN NWDAT) 1553
23.15.11 Interrupt Pending Registers (CAN INTPND) 1553
23.15.12 Message Valid Registers (CAN MSGVAL) 1554
23.15.13 Interrupt Multiplexer Registers (CAN INTMUX) 1554
23.15.14 IF1 and IF2 Command Registers (CAN IF1CMD, CAN IF2CMD) 1555
23.15.15 IF1 and IF2 Mask Registers (CAN IF1MSK, CAN IF2MSK) 1557
23.15.16 IF1 and F2 Arbitration Registers (CAN IF1ARB, CAN IF2ARB) 1558
23.15.17 IF1 and IF2 Message Control Registers (CAN IF1MCTL, CAN IF2MCTL) 1559
23.15.18 IF1 and IF2 Data A and Data B Registers (CAN IF1DATA/DATB, CAN IF2DATA/DATB) 1561
23.15.19 IF3 Observation Register (CAN IF3OBS) 1562
23.15.20 IF3 Mask Register (CAN IF3MSK) 1563
23.15.21 IF3 Arbitration Register (CAN IF3ARB) 1563
23.15.22 IF3 Message Control Register (CAN IF3MCTL) 1564
23.15.23 IF3 Data A and Data B Registers (CAN IF3DATA/DATB) 1565
23.15.24 IF3 Update Enable Registers (CAN IF3UPD) 1566

24 Cortex-M3 Processor 1567

24.1 Overview 1568
24.2 Block Diagram 1568
24.3 Overview 1569
24.3.1 System-Level Interface 1569
24.3.2 System Component Details 1569
24.4 Programming Model 1570
24.4.1 Processor Mode and Privilege Levels for Software Execution 1570
24.4.2 Stacks 1570
24.4.3 Register Map 1570
24.4.4 Register Descriptions 1572
24.4.5 Exceptions and Interrupts 1578
24.4.6 Data Types 1578
24.5 Memory Model 1578
24.6 Memory Regions, Types and Attributes 1579
24.6.1 Memory System Ordering of Memory Accesses 1579
24.6.2 Behavior of Memory Accesses 1579
24.6.3 Software Ordering of Memory Accesses 1580
24.6.4 Bit-Banding 1581
24.6.5 Data Storage 1582
24.6.6 Synchronization Primitives 1583
24.7 Exception Model 1584
24.7.1 Exception States 1584
24.7.2 Exception Types 1584
24.7.3 Exception Handlers 1588
24.7.4 Vector Table 1588
24.7.5 Exception Priorities 1589
24.7.6 Interrupt Priority Grouping 1590
24.7.7 Exception Entry and Return 1590
24.8 Fault Handling 1592
24.8.1 Fault Types 1592

Copyright © 2012–2019, Texas Instruments Incorporated

24.8.2 Fault Escalation and Hard Faults 1593
24.8.3 Fault Status Registers and Fault Address Registers 1593
24.8.4 Lockup 1593
24.9 Power Management 1593
24.9.1 Entering Sleep Modes 1594
24.9.2 Wake Up from Sleep Mode 1594
24.10 Instruction Set Summary 1595
25 Cortex-M3 Peripherals 1598
25.1 Overview 1599
25.2 Functional Description 1599
25.2.1 System Timer (SysTick) 1599
25.2.2 Nested Vectored Interrupt Controller (NVIC) 1600
25.2.3 System Control Block (SCB) 1601
25.2.4 Memory Protection Unit (MPU) 1601
25.3 Register Map 1606
25.4 System Timer (SysTick) Register Descriptions 1609
25.4.1 SysTick Control and Status Register (STCTRL), offset 0x010 1609
25.4.2 SysTick Reload Value Register (STRELOAD), offset 0x014 1610
25.4.3 SysTick Current Value Register (STCURRENT), offset 0x018 1610
25.5 NVIC Register Descriptions 1611
25.5.1 Interrupt 0-31 Set Enable (EN0) Register, offset 0x100 1611
25.5.2 Interrupt 32-63 Set Enable 1 (EN1), offset 0x104 1612
25.5.3 Interrupt 64-95 Set Enable 2 (EN2), offset 0x108 1612
25.5.4 Interrupt 96-127 Set Enable 3 (EN3), offset 0x10C 1612
25.5.5 Interrupt 128-133 Set Enable 4 (EN4), offset 0x110 1613
25.5.6 Interrupt 0-31 Clear Enable (DIS0) Register, offset 0x180 1614
25.5.7 Interrupt 32-63 Clear Enable (DIS1) Register, offset 0x184 1614
25.5.8 Interrupt 64-95 Clear Enable (DIS2) Register, offset 0x188 1615
25.5.9 Interrupt 96-127 Clear Enable (DIS3) Register, offset 0x18C 1616
25.5.10 Interrupt 128-133 Clear Enable (DIS4) Register, offset 0x190 1617
25.5.11 Interrupt 0-31 Set Pending (PEND0) Register, offset 0x200 1617
25.5.12 Interrupt 32-63 Set Pending (PEND1) Register, offset 0x204 1618
25.5.13 Interrupt 64-95 Set Pending (PEND2) Register, offset 0x208 1618
25.5.14 Interrupt 96-127 Set Pending (PEND3) Register, offset 0x20C 1618
25.5.15 Interrupt 128-133 Set Pending (PEND4) Register, offset 0x210 1619
25.5.16 Interrupt 0-31 Clear Pending (UNPEND0) Register, offset 0x280 1620
25.5.17 Interrupt 32-63 Clear Pending (UNPEND1) Register, offset 0x284 1620
25.5.18 Interrupt 64-95 Clear Pending (UNPEND2) Register, offset 0x288 1621
25.5.19 Interrupt 96-127 Clear Pending (UNPEND3) Register, offset 0x28C 1622
25.5.20 Interrupt 128-133 Clear Pending (UNPEND4) Register, offset 0x290 1623
25.5.21 Interrupt 0-31 Active Bit (ACTIVE0) Register, offset 0x300 1623
25.5.22 Interrupt 32-63 Active Bit (ACTIVE1) Register, offset 0x304 1624
25.5.23 Interrupt 64-95 Active Bit (ACTIVE2) Register, offset 0x308 1624
25.5.24 Interrupt 96-127 Active Bit (ACTIVE3) Register, offset 0x30C 1624
25.5.25 Interrupt 128-133 Active Bit (ACTIVE4) Register, offset 0x310 1625
25.5.26 Interrupt 0-133 Priority (PRI0-PRI33) Registers, offset 0x400-0x484 1626
25.5.27 Software Trigger Interrupt (SWTRIG) Register, offset 0xF00 1626
25.6 System Control Block (SCB) Register Descriptions 1628
25.6.1 Auxiliary Control (ACTLR) Register, offset 0x008 1628
25.6.2 CPU ID Base (CPUID) Register, offset 0xD00 1629
25.6.3 Interrupt Control and State (INTCTRL) Register, offset 0xD04 1630
25.6.4 Vector Table Offset (VTABLE) Register, offset 0xD08 1633
25.6.5 Application Interrupt and Reset Control (APINT) Register, offset 0xD0C 1634

Copyright © 2012–2019, Texas Instruments Incorporated

25.6.6 System Control (SYSCTRL) Register, offset 0xD10 1636
25.6.7 Configuration and Control (CFGCTRL) Register, offset 0xD14 1637
25.6.8 System Handler Priority 1 (SYSPRI1) Register, offset 0xD18 1638
25.6.9 System Handler Priority 2 (SYSPRI2) Register, offset 0xD1C 1639
25.6.10 System Handler Priority 3 (SYSPRI3) Register, offset 0xD20 1639
25.6.11 System Handler Control and State (SYSHNDCTRL) Register, offset 0xD24 1640
25.6.12 Configurable Fault Status (FAULTSTAT) Register, offset 0xD28 1643
25.6.13 Hard Fault Status (HFAULTSTAT) Register, offset 0xD2C 1647
25.6.14 Memory Management Fault Address (MMADDR) Register, offset 0xD34 1648
25.6.15 Bus Fault Address (FAULTADDR) Register, offset 0xD38 1648
25.7 Memory Protection Unit (MPU) Register Descriptions 1649
25.7.1 MPU Type (MPUTYPE) Register, offset 0xD90 1649
25.7.2 MPU Control (MPUCTRL) Register, offset 0xD94 1650
25.7.3 MPU Region Number (MPUNUMBER) Register, offset 0xD98 1651
25.7.4 MPU Region Base Address (MPUBASE) Register, Offset 0xD9c-0xDB4 1651
25.7.5 MPU Region Attribute and Size (MPUATTR) Register, offset 0xDA0-DB8 1652
Revision History **1654**

Copyright © 2012–2019, Texas Instruments Incorporated

List of Figures

1-1. Resets Connectivity 88
1-2. Master and Control Subsystem WIR Mode Flow 94
1-3. Master Subsystem NMI Sources and MNMIWD 98
1-4. PIE Interrupts Multiplexing 101
1-5. CPU Level Interrupt Handling 103
1-6. Control Subsystem C28x Processor After Reset Flow 105
1-7. PIE Interrupt Sources and External Interrupts XINT1/XINT2/XINT3 106
1-8. Multiplexed Interrupt Request Flow 108
1-9. Control Subsystem NMI Sources and CNMIWD 117
1-10. Missing Clock Detection Logic 122
1-11. Master Subsystem Clocks and Low Power Mode Configuration 128
1-12. Control Subsystem Clocks and Low Power Mode Configuration 131
1-13. Control Subsystem Peripherals Clocking 133
1-14. CPU-Timers 136
1-15. CPU-Timer Interrupts Signals and Output Signal 136
1-16. TIMERxTIM Register (x = 0, 1, 2) 137
1-17. TIMERxTIMH Register (x = 0, 1, 2) 137
1-18. TIMERxPRD Register (x = 0, 1, 2) 138
1-19. TIMERxPRDH Register (x = 0, 1, 2) 138
1-20. TIMERxTCR Register (x = 0, 1, 2) 138
1-21. TIMERxTPR Register (x = 0, 1, 2) 139
1-22. TIMERxTPRH Register (x = 0, 1, 2) 140
1-23. CSM Password Match Flow 150
1-24. ECSL Password Match Flow 153
1-25. Messaging with IPC Flags and Interrupts 158
1-26. Flash Pump Allocation for Different States of Flash Pump Semaphore 162
1-27. Mastership of Clock Configuration Registers for Different States of Clock Configuration Semaphore 163
1-28. Device Identification 0 (DID0) Register 173
1-29. Device Identification 1 (DID1) Register 173
1-30. Device Configuration 1 (DC1) Register 174
1-31. Device Configuration 2 (DC2) Register 175
1-32. Device Configuration 4 (DC4) Register 177
1-33. Device Configuration 6 (DC6) Register 179
1-34. Device Configuration 10 (DC10) Register 179
1-35. Device Configuration 7 (DC7) Register 180
1-36. Master Subsystem Configuration (MCNF) Register 180
1-37. Serial Port Loop Back Control (SERPLOOP) Register 181
1-38. Master Subystem: ACIB Status (MCIBSTATUS) Register 181
1-39. C28 Device Part ID (PARTID) Register 182
1-40. C28 Revision ID (REVID) Register 182
1-41. Control Subsystem Device Configuration (DEVICECNF) Register 183
1-42. Control Subsystem Peripheral Configuration 0 (CCNF0) Register 183
1-43. Control Subsystem Peripheral Configuration 1 (CCNF1) Register 184
1-44. Control Subsystem Peripheral Configuration 2 (CCNF2) Register 185
1-45. Control Subsystem Peripheral Configuration 3 (CCNF3) Register 186
1-46. Control Subsystem Peripheral Configuration 4 (CCNF4) Register 187
1-47. Master Subsystem Memory Configuration (MEMCNF) Register 187

Copyright © 2012–2019, Texas Instruments Incorporated

1-48. Subsystem Reset Configuration/Control (CRESCNF) Register 188
1-49. Control Subsystem Reset Status (CRESSTS) Register 188
1-50. Master Reset Cause (MRESC) Register 190
1-51. C28 Reset Cause Register (CRESC) Register 191
1-52. Software Reset Control 0 (SRCR0) Register 192
1-53. Software Reset Control 1 (SRCR1) Register 193
1-54. Software Reset Control 2 (SRCR2) Register 194
1-55. Software Reset Control 3 (SRCR3) Register 195
1-56. Master Subsystem Wait-In-Reset (MWIR) Register 197
1-57. C28 Wait-In-Reset (CWIR) Register 197
1-58. M3NMI Configuration (MNMICFG) Register 198
1-59. M3NMI Flag (MNMIFLG) Register 199
1-60. M3NMI Flag Clear (MNMIFLGCLR) Register 200
1-61. M3NMI Flag Force (MNMIFLGFRC) Register 201
1-62. M3NMI Watchdog Counter (MNMIWDCNT) Register 202
1-63. M3NMI Watchdog Period (MNMIWDPRD) Register 202
1-64. C28 NMI Configuration (CNMICFG) Register 203
1-65. C28 NMI Flag (CNMIFLG) Register 203
1-66. C28 NMI Flag Clear (CNMIFLGCLR) Register 205
1-67. C28 NMI Flag Force (CNMIFLGFRC) Register 206
1-68. C28 NMI Watchdog Counter (CNMIWDCNT) Register 206
1-69. C28 NMI Watchdog Period (CNMIWDPRD) Register 207
1-70. PIE, Control (PIECTRL) Register 207
1-71. PIE, Acknowledge (PIEACK) Register 207
1-72. PIE, INTx Group Enable Register (PIEIERx) (x = 1 to 12) 208
1-73. PIE, INTx Group Flag Register (PIEIFRx) (x = 1 to 12) 209
1-74. CPU Interrupt Flag Register (IFR) 210
1-75. CPU Interrupt Enable Register (IER) 212
1-76. Debug Interrupt Enable Register (DBGIER) 213
1-77. C28 External Interrupt 1 Configuration Register (XINT1CR) 215
1-78. C28 External Interrupt 2 Configuration Register (XINT2CR) 215
1-79. C28 External Interrupt 3 Configuration Register (XINT3CR) 215
1-80. C28 External Interrupt 1 Counter Register (XINT1CTR) 216
1-81. C28 External Interrupt 2 Counter Register (XINT2CTR) 216
1-82. C28 External Interrupt 3 Counter Register (XINT3CTR) 217
1-83. System PLL Configuration (SYSPLLCTL) Register 217
1-84. Control Subsystem Clock Disable (CCLKOFF) Register 218
1-85. M3 Configuration Write Allow (MWRALLOW) Register 218
1-86. M3 Configuration Lock (MLOCK) Register 218
1-87. Missing Clock Status (MCLKSTS) Register 219
1-88. Missing Clock Force (MCLKFRCCLR) Register 219
1-89. Missing Clock Enable (MCLKEN) Register 220
1-90. Missing Clock Reference Limit (MCLKLIMIT) Register 220
1-91. C28 USER_SWREG1 Register 221
1-92. C28_USER_SWREG2 Register 221
1-93. System PLL Multiplier (SYSPLLMULT) Register 222
1-94. System Clock Divider (SYSDIVSEL) Register 222
1-95. System PLL Lock Status (SYSPLLSTS) Register 223
1-96. Master Subsystem Clock Divider (M3SSDIVSEL) Register 223

Copyright © 2012–2019, Texas Instruments Incorporated

1-97. XPLL CLKOUT Control (XPLLCLKCFG) Register 224
1-98. USB PLL Configuration (UPLLCTL) Register 224
1-99. USB PLL Multiplier (UPLLMULT) Register 225
1-100. USB PLL Lock Status (UPLLSTS) Register 226
1-101. Bit Clock Source Selection for CAN0 (CAN0BCLKSEL) Register 226
1-102. Bit Clock Source Selection for CAN1 (CAN1BCLKSEL) Register 227
1-103. Run Mode Clock Configuration (RCC) Register 227
1-104. Master GPIO High Performance Bus Control (GPIOHBCTL) Register 227
1-105. Run Mode Clock Gating Control Register 0 (RCGC0) 228
1-106. Sleep Mode Clock Gating Control Register 0 (SCGC0) 229
1-107. Deep Sleep Mode Clock Gating Control Register 0 (DCGC0) 229
1-108. Run Mode Clock Gating Control Register 1 (RCGC1) 230
1-109. Sleep Mode Clock Gating Control Register 1 (SCGC1) 231
1-110. Deep Sleep Mode Clock Gating Control Register 1 (DCGC1) 233
1-111. Run Mode Clock Gating Control Register 2 (RCGC2) 234
1-112. Sleep Mode Clock Gating Control Register 2 (SCGC2) 236
1-113. Deep Sleep Mode Clock Gating Control Register 2 (DCGC2) 238
1-114. Run Mode Clock Gating Control Register 3 (RCGC3) 239
1-115. Sleep Mode Clock Gating Control Register 3 (SCGC3) 239
1-116. Deep Sleep Mode Clock Gating Control Register 3 (DCGC3) 240
1-117. Deep Sleep Clock Configuration (DSLPCLKCFG) Register 241
1-118. C28 CPU Timer 2 Clock Configuration (CLKCTL) Register 242
1-119. Peripheral Clock Control Register 0 (PCLKCR0) 242
1-120. Peripheral Clock Control Register 1 (PCLKCR1) 243
1-121. Peripheral Clock Control Register 2 (PCLKCR2) 244
1-122. Peripheral Clock Control Register 3 (PCLKCR3) 245
1-123. High-Speed Clock Prescaler (CHISPCP) Register 246
1-124. Low-Speed Clock Prescaler (CLOSPCP) Register 246
1-125. C28 XCLKOUT Divider Register (CXCLK) 247
1-126. Z1_CSMKEY0 Register 248
1-127. Z1_CSMKEY1 Register 248
1-128. Z1_CSMKEY2 Register 248
1-129. Z1_CSMKEY3 Register 248
1-130. Z1_ECSLKEY0 Register 249
1-131. Z1_ECSLKEY1 Register 249
1-132. Z2_CSMKEY0 Register 249
1-133. Z2_CSMKEY1 Register 250
1-134. Z2_CSMKEY2 Register 250
1-135. Z2_CSMKEY3 Register 250
1-136. Z2_ECSLKEY0 Register 250
1-137. Z2_ECSLKEY1 Register 251
1-138. Z1_CSMCR Register 251
1-139. Z2_CSMCR Register 252
1-140. Z1_GRABSECTR Register 254
1-141. Z1_GRABRAMR Register 256
1-142. Z2_GRABSECTR Register 256
1-143. Z2_GRABRAMR Register 258
1-144. Z1_EXEONLYR Register 260
1-145. Z2_EXEONLYR Register 261

Copyright © 2012–2019, Texas Instruments Incorporated

1-146. OTPSECLOCK Register 262
1-147. Z1_CSMKEY0 Register 264
1-148. CSMKEY1 Register 264
1-149. CSMKEY2 Register 264
1-150. CSMKEY3 Register 264
1-151. CSMCR Register 265
1-152. ECSLKEY0 Register 266
1-153. ECSLKEY1 Register 266
1-154. EXEONLYR Register 267
1-155. µCRCCONFIG Register 269
1-156. µCRCCONTROL Register 269
1-157. µCRCRES Register 269
1-158. M3 to C28 IPC Set (MTOCIPCSET) Register 270
1-159. M3 to C28 IPC Clear (MTOCIPCCLR) Register 272
1-160. M3 to C28 Core Flag (MTOCIPCFLG) Register 274
1-161. M3 to C28 Core IPC Acknowledge (CTOMIPCACK) Register 276
1-162. C28 to M3 Core IPC Status (CTOMIPCSTS) Register 278
1-163. M3 Flash Semaphore Register 280
1-164. M3 Clock Semaphore Register 281
1-165. CTOMIPCSET Register 281
1-166. CTOMIPCCLR Register 283
1-167. CTOMIPCFLG Register 285
1-168. MTOCIPCACK Register 287
1-169. MTOCIPCSTS Register 289
1-170. C28 Flash Semaphore Register 291
1-171. C28 Clock Semaphore Register 292
1-172. MIPCCOUNTERL Register 293
1-173. MIPCCOUNTERH Register 293
1-174. CTOMIPCCOM Register 293
1-175. CTOMIPCADDR Register 294
1-176. CTOMIPCDATAW Register 294
1-177. CTOMIPCDATAR Register 294
1-178. MTOCIPCCOM Register 294
1-179. MTOCIPCADDR Register 296
1-180. MTOCIPCDATAW Register 296
1-181. MTOCIPCDATAR Register 296
1-182. CTOMIPCBOOTSTS Register 296
1-183. MTOCIPCBOOTMODERegister 297
2-1. GPTM Block Diagram 299
2-2. Timer Daisy Chain 302
2-3. Edge-Count Mode Example 303
2-4. 16-Bit Input Edge-Time Mode Example 304
2-5. 16-Bit PWM Mode Example 305
2-6. GPTM Configuration (GPTMCFG) Register 309
2-7. GPTM Timer A Mode (GPTMTAMR) Register 310
2-8. GPTM Timer B Mode (GPTMTBMR) Register 311
2-9. GPTM Control (GPTMCTL) Register 312
2-10. GPTM Interrupt Mask (GPTMIMR) Register 313
2-11. GPTM Raw Interrupt Status (GPTMRIS) Register 314

Copyright © 2012–2019, Texas Instruments Incorporated

2-12. GPTM Masked Interrupt Status (GPTMMIS) Register 315
2-13. GPTM Interrupt Clear (GPTMICR) Register 317
2-14. GPTM Timer A Interval Load (GPTMTAILR) Register 318
2-15. GPTM Timer A Interval Load (GPTMTBILR) Register 318
2-16. GPTM Timer A Match (GPTMTAMATCHR) Register 319
2-17. GPTM Timer B Match (GPTMTBMATCHR) Register 319
2-18. GPTM Timer A Prescale (GPTMTAPR) Register 319
2-19. GPTM Timer A Prescale (GPTMTBPR) Register 320
2-20. GPTM Timer A Prescale Match (GPTMTAPMR) Register 320
2-21. GPTM Timer B Prescale Match (GPTMTBPMR) Register 321
2-22. GPTM Timer A (GPTMTAR) Register 321
2-23. GPTM Timer B (GPTMTBR) Register 322
2-24. GPTM Timer A Value (GPTMTAV) Register 322
2-25. GPTM Timer B Value (GPTMTBV) Register 323
3-1. Watchdog Timer Module Block Diagram 325
3-2. Watchdog Load (WDTLOAD) Register 327
3-3. Watchdog Value (WDTVALUE) Register 328
3-4. Watchdog Control (WDTCTL) Register 329
3-5. Watchdog Interrupt Clear (WDTICR) Register 329
3-6. Watchdog Raw Interrupt Status (WDTRIS) Register 330
3-7. Watchdog Masked Interrupt Status (WDTMIS) Register 330
3-8. Watchdog Test (WDTTEST) Register 331
3-9. Watchdog Lock (WDTLOCK) Register 331
3-10. Watchdog Peripheral Identification 4 (WDTPeriphID4) Register 332
3-11. Watchdog Peripheral Identification 5 (WDTPeriphID5) Register 332
3-12. Watchdog Peripheral Identification 6 (WDTPeriphID6) Register 332
3-13. Watchdog Peripheral Identification 7 (WDTPeriphID7) Register 333
3-14. Watchdog Peripheral Identification 0 (WDTPeriphID0) Register 333
3-15. Watchdog Peripheral Identification 1 (WDTPeriphID1) Register 333
3-16. Watchdog Peripheral Identification 2 (WDTPeriphID2) Register 334
3-17. Watchdog Peripheral Identification 3 (WDTPeriphID3) Register 334
3-18. Watchdog PrimeCell Identification 0 (WDTPCellID0) Register 334
3-19. Watchdog PrimeCell Identification 1 (WDTPCellID1) Register 335
3-20. Watchdog PrimeCell Identification 2 (WDTPCellID2) Register 335
3-21. Watchdog PrimeCell Identification 3 (WDTPCellID3) Register 335
4-1. Digital I/O Pads 342
4-2. GPIODATA Write Example 343
4-3. GPIODATA Read Example 343
4-4. GPIO Data (GPIODATA) Register 348
4-5. GPIO Direction (GPIODIR) Register 349
4-6. GPIO Interrupt Sense (GPIOIS) Register 349
4-7. GPIO Interrupt Both Edges (GPIOIBE) Register 350
4-8. GPIO Interrupt Event (GPIOIEV) Register 350
4-9. GPIO Interrupt Mask (GPIOIM) Register 351
4-10. GPIO Raw Interrupt Status (GPIORIS) Register 351
4-11. GPIO Masked Interrupt Status (GPIOMIS) Register 352
4-12. GPIO Interrupt Clear (GPIOICR) Register 352
4-13. GPIO Alternate Function Select (GPIOAFSEL) Register 353
4-14. GPIO Open Drain Select (GPIOODR) Register 354

Copyright © 2012–2019, Texas Instruments Incorporated

4-15. GPIO Pull-Up Select (GPIOPUR) Register 354
4-16. GPIO Digital Enable (GPIODEN) Register 355
4-17. GPIO Lock (GPIOLOCK) Register 356
4-18. GPIO Commit (GPIOCR) Register 357
4-19. GPIO Analog Mode Select (GPIOAMSEL) Register 357
4-20. GPIO Port Control (GPIOPCTL) Register 358
4-21. GPIO Alternate Peripheral Select (GPIOAPSEL) Register 359
4-22. GPIO Core Select (GPIOCSEL) Register 360
4-23. GPIO Peripheral Identification 4 (GPIOPeriphID4) Register 361
4-24. GPIO Peripheral Identification 5 (GPIOPeriphID5) Register 362
4-25. GPIO Peripheral Identification 6 (GPIOPeriphID6) Register 363
4-26. GPIO Peripheral Identification 7 (GPIOPeriphID7) Register 364
4-27. GPIO Peripheral Identification 0 (GPIOPeriphID0) Register 365
4-28. GPIO Peripheral Identification 1 (GPIOPeriphID1) Register 365
4-29. GPIO Peripheral Identification 2 (GPIOPeriphID2) Register 366
4-30. GPIO Peripheral Identification 3 (GPIOPeriphID3) Register 366
4-31. GPIO PrimeCell Identification 0 (GPIOPCellID0) Register 367
4-32. GPIO PrimeCell Identification 1 (GPIOPCellID1) Register 367
4-33. GPIO PrimeCell Identification 2 (GPIOPCellID2) Register 367
4-34. GPIO PrimeCell Identification 3 (GPIOPCellID3) Register 368
4-35. GPIO0 to GPIO31 Multiplexing Diagram 370
4-36. GPIO32, GPIO33 Multiplexing Diagram 371
4-37. GPIO34, GPIO135 Multiplexing Diagram 372
4-38. Analog/GPIO Multiplexing 373
4-39. GPIO MUX-to-Trip Input Connectivity 374
4-40. Input Qualification Using a Sampling Window 380
4-41. Input Qualifier Clock Cycles 383
4-42. GPIO Port A MUX 1 (GPAMUX1) Register 391
4-43. GPIO Port A MUX 2 (GPAMUX2) Register 393
4-44. GPIO Port B MUX 1 (GPBMUX1) Register 395
4-45. GPIO Port B MUX 2 (GPBMUX2) Register 397
4-46. GPIO Port C MUX 1 (GPCMUX1) Register 399
4-47. GPIO Port E MUX 1 (GPEMUX1) Register 400
4-48. Analog I/O MUX 1 (AIOMUX1) Register 401
4-49. Analog I/O MUX 2 (AIOMUX2) Register 402
4-50. GPIO Port A Qualification Control (GPACTRL) Register 403
4-51. GPIO Port B Qualification Control (GPBCTRL) Register 404
4-52. GPIO Port C Qualification Control (GPCCTRL) Register 405
4-53. GPIO Port E Qualification Control (GPECTRL) Register 405
4-54. GPIO Port A Qualification Select 1 (GPAQSEL1) Register 406
4-55. GPIO Port A Qualification Select 2 (GPAQSEL2) Register 406
4-56. GPIO Port B Qualification Select 1 (GPBQSEL1) Register 407
4-57. GPIO Port B Qualification Select 2 (GPBQSEL2) Register 407
4-58. GPIO Port C Qualification Select 1 (GPCQSEL1) Register 408
4-59. GPIO Port E Qualification Select 1 (GPEQSEL1) Register 409
4-60. GPIO Port A Direction (GPADIR) Register 409
4-61. GPIO Port B Direction (GPBDIR) Register 410
4-62. GPIO Port C Direction (GPCDIR) Register 410
4-63. GPIO Port E Direction (GPEDIR) Register 411

Copyright © 2012–2019, Texas Instruments Incorporated

4-64. GPIO Port E Pullup Disable (GPEPUD) 412
4-65. Analog I/O DIR (AIODIR) Register 412
4-66. GPIO Port A Data (GPADAT) Register 413
4-67. GPIO Port B Data (GPBDAT) Register 414
4-68. GPIO Port C Data (GPCDAT) Register 415
4-69. GPIO Port E Data (GPEDAT) Register 416
4-70. Analog I/O DAT (AIODAT) Register 417
4-71. GPIO Port A Set, Clear and Toggle (GPASET, GPACLEAR, GPATOGGLE) Registers 418
4-72. GPIO Port B Set, Clear and Toggle (GPBSET, GPBCLEAR, GPBTOGGLE) Registers 419
4-73. GPIO Port C Set, Clear and Toggle (GPCSET, GPCCLEAR, GPCTOGGLE) Registers 420
4-74. GPIO Port E Set, Clear and Toggle (GPESET, GPECLEAR, GPETOGGLE) Registers 421
4-75. Analog I/O Toggle (AIOSET, AIOCLEAR, AIOTOGGLE) Register 422
4-76. GPIO Trip Input Select Register (GPTRIPxSEL) 423
4-77. GPIO Low Power Mode Wakeup Select 1 (GPIOLPMSEL1) Register 424
4-78. GPIO Low Power Mode Wakeup Select 2 (GPIOLPMSEL2) Register 424
5-1. RAM Control 426
5-2. Shared RAM (Dedicated to Subsystem) 427
5-3. Shared RAM (Shared between Subsystems) 427
5-4. Cx DEDRAM Configuration Register 1 (CxDRCR1) 437
5-5. Cx SHRAM Configuration Register 1 (CxSRCR1) 438
5-6. Sx SHRAM Master Select Register (MSxMSEL) 439
5-7. M3 Sx SHRAM Configuration Register 1 (MSxSRCR1) 440
5-8. M3 Sx SHRAM Configuration Register 2 (MSxSRCR2) 442
5-9. M3TOC28_MSG_RAM Configuration Register (MTOCMSGRCR) 444
5-10. Cx RAM Test and Initialization Register 1 (CxRTESTINIT1) 445
5-11. M3 Sx RAM Test and Initialization Register 1 (MSxRTESTINIT1) 446
5-12. MTOC_MSG_RAM Test and Initialization Register (MTOCRTESTINIT) 448
5-13. Cx RAM INITDONE Register 1 (CxRINITDONE1) 449
5-14. M3 Sx RAM INITDONE Register 1 (MSxRINITDONE1) 450
5-15. MTOC_MSG_RAM INITDONE Register (MTOCRINITDONE) 452
5-16. M3 CPU Uncorrectable Write Error Address Register (MCUNCWEADDR) 452
5-17. M3 µDMA Uncorrectable Write Error Address Register (MDUNCWEADDR) 452
5-18. M3 CPU Uncorrectable Read Error Address Register (MCUNCREADDR) 453
5-19. M3 µDMA Uncorrectable Read Error Address Register (MDUNCREADDR) 453
5-20. M3 CPU Corrected Read Error Address Register (MCPUCREADDR) 454
5-21. M3 µDMA Corrected Read Error Address Register (MDMACREADDR) 454
5-22. M3 Uncorrectable Error Flag Register (MUEFLG) 455
5-23. M3 Uncorrectable Error Force Register (MUEFRC) 456
5-24. M3 Uncorrectable Error Flag Clear Register (MUECLR) 457
5-25. M3 Corrected Error Counter Register (MCECNTR) 457
5-26. M3 Corrected Error Threshold Register (MCETRES) 458
5-27. M3 Corrected Error Threshold Exceeded Flag Register (MCEFLG) 458
5-28. M3 Corrected Error Threshold Exceeded Force Register (MCEFRC) 458
5-29. M3 Corrected Error Threshold Exceeded Flag Clear Register (MCECLR) 459
5-30. M3 Single Error Interrupt Enable Register (MCEIE) 459
5-31. Non-Master Access Violation Flag Register (MNMAVFLG) 460
5-32. Non-Master Access Violation Flag Clear Register (MNMAVCLR) 460
5-33. Master Access Violation Flag Register (MMAVFLG) 461
5-34. Master Access Violation Flag Clear Register (MMAVCLR) 462

Copyright © 2012–2019, Texas Instruments Incorporated

5-35. Non-Master CPU Write Access Violation Address Register (MNMWRAVADDR) 462
5-36. Non-Master DMA Write Access Violation Address Register (MNMDMAWRAVADDR) 463
5-37. Non-Master CPU Fetch Access Violation Address Register (MNMFAVADDR) 463
5-38. Master CPU Write Access Violation Address Register (MMWRAVADDR) 463
5-39. Master DMA Write Access Violation Address Register (MMDMAWRAVADDR) 464
5-40. Master CPU Fetch Access Violation Address Register (MMFAVADDR) 464
5-41. Lx DEDRAM Configuration Register 1 (LxDRCR1) 465
5-42. Lx SHRAM Configuration Register 1 (LxSRCR1) 466
5-43. C28x Sx SHRAM Master Select Register (CSxMSEL) 467
5-44. C28x Sx SHRAM Configuration Register 1 (CSxSRCR1) 468
5-45. C28x Sx SHRAM Configuration Register 2 (CSxSRCR2) 469
5-46. C28TOC28_MSG_RAM Configuration Register (CTOMMSGRCR) 471
5-47. M0, M1 and C28T0C28_MSG_RAM Test and Initialization Register (C28RTESTINIT) 472
5-48. Lx RAM Test and Initialization Register 1 (CLxRTESTINIT1) 473
5-49. C28x Sx RAM Test and Initialization Register 1 (CSxRTESTINIT1) 474
5-50. M0, M1 and C28T0M3_MSG_RAM INIT Done Register (C28RINITDONE) 476
5-51. C28x Lx RAM_INIT_DONE Register 1 (CLxRINITDONE1) 477
5-52. C28x Sx RAM_INIT_DONE Register 1 (CSxRINITDONE1) 478
5-53. C28x CPU Uncorrectable Read Error Address Register (CCUNCREADDR) 480
5-54. C28x DMA Uncorrectable Read Error Address Register (CDUNCREADDR) 480
5-55. C28x CPU Corrected Read Error Address Register (CCPUCREADDR) 480
5-56. C28x DMA Corrected Read Error Address Register (CDMACREADDR) 481
5-57. C28x Uncorrectable Error Flag Register (CUEFLG) 481
5-58. C28x Uncorrectable Error Force Register (CUEFRC) 482
5-59. C28x Uncorrectable Error Flag Clear Register (CUECLR) 482
5-60. C28x Corrected Error Counter Register (CCECNTR) 483
5-61. C28x Corrected Error Threshold Register (CCETRES) 483
5-62. C28x Corrected Error Threshold Exceeded Flag Register (CCEFLG) 484
5-63. C28x Corrected Error Threshold Exceeded Force Register (CCEFRC) 484
5-64. C28x Corrected Error Threshold Exceeded Flag Clear Register (CCECLR) 485
5-65. C28x Single Error Interrupt Enable Register (CCEIE) 485
5-66. Non-Master Access Violation Flag Register (CNMAVFLG) 486
5-67. Non-Master Access Violation Force Register (CNMAVFRC) 487
5-68. Non-Master Access Violation Flag Clear Register (CNMAVCLR) 487
5-69. Master Access Violation Flag Register (CMAVFLG) 488
5-70. Master Access Violation Force Register (CMAVFRC) 488
5-71. Master Access Violation Flag Clear Register (CMAVCLR) 490
5-72. Non-Master CPU Write Access Violation Address Register (CNMWRAVADDR) 491
5-73. Non-Master DMA Write Access Violation Address Register (CNMDMAWRAVADDR) 491
5-74. Non-Master CPU Fetch Access Violation Address Register (CNMFAVADDR) 491
5-75. Master CPU Write Access Violation Address Register (CMWRAVADDR) 492
5-76. Master DMA Write Access Violation Address Register (CMDMAWRAVADDR) 492
5-77. Master CPU Fetch Access Violation Address Register (CMFAVADDR) 492
5-78. FMC Interface with Core, Bank and Pump 495
5-79. Flash Cache Mode 498
5-80. Flash Prefetch Mode 501
5-81. ECC Logic Inputs and Outputs 504
5-82. Flash Read Control Register (FRDCNTL) 512
5-83. Flash Read Margin Control Register (FSPRD) 512

Copyright © 2012–2019, Texas Instruments Incorporated

5-84. Flash Bank Access Control Register (FBAC) 513
5-85. Flash Bank Fallback Power Register (FBFALLBACK) 513
5-86. Flash Bank Pump Control Register (FBPRDY) 514
5-87. Flash Bank Pump Control Register 1 (FPAC1) 514
5-88. Flash Bank Pump Control Register 2 (FPAC2) 515
5-89. Flash Module Access Control Register (FMAC) 515
5-90. SECZONEREQUEST(SEM) Register 516
5-91. Flash Read Interface Control Register (FRD_INTF_CTRL) 517
5-92. ECC Enable Register (ECC_Enable) 518
5-93. Single Error Address Register (SINGLE_ERR_ADDR) 518
5-94. Uncorrectable Error Address Register (UNC_ERR_ADDR) 518
5-95. Error Status Register (ERR_STATUS) 519
5-96. Error Position Register (ERR_POS) 519
5-97. Error Status Clear Register (ERR_STATUS_CLR) 520
5-98. Error Counter Register (ERR_CNT) 520
5-99. Error Threshold Register (ERR_THRESHOLD) 521
5-100. Error Interrupt Flag Register (ERR_INTFLG) 521
5-101. Error Interrupt Flag Clear Register (ERR_INTCLR) 522
5-102. Data High Test Register (FDATAH_TEST) 522
5-103. Data Low Test Register (FDATAL_TEST) 522
5-104. ECC Test Address Register (FADDR_TEST) 523
5-105. ECC Test Register (FECC_TEST) 523
5-106. ECC Control Register (FECC_CTRL) 523
5-107. Test Data Out High Register (FECC_FOUTH_TEST) 524
5-108. Test Data Out Low Register (FECC_FOUTL_TEST) 524
5-109. ECC Status Register (FECC_STATUS) 524
5-110. Flash Read Control Register (FRDCNTL) 525
5-111. Flash Read Margin Control Register (FSPRD) 525
5-112. Flash Bank Access Control Register (FBAC) 526
5-113. Flash Bank Fallback Power Register (FBFALLBACK) 526
5-114. Flash Bank Pump Control Register (FBPRDY) 527
5-115. Flash Bank Pump Control Register 1 (FPAC1) 527
5-116. Flash Bank Pump Control Register 2 (FPAC2) 528
5-117. Flash Module Access Control Register (FMAC) 528
5-118. Flash Read Interface Control Register (FRD_INTF_CTRL) 528
5-119. ECC Enable Register (ECC_ENABLE) 529
5-120. SIngle Error Address Register (SINGLE_ERR_ADDR) 529
5-121. Uncorrectable Error Address Register (UNC_ERR_ADDR) 529
5-122. Error Status Register (ERR_STATUS) 530
5-123. Error Position Register (ERR_POS) 530
5-124. Error Status Clear Register (ERR_STATUS_CLR) 531
5-125. Error Counter Register (ERR_CNT) 531
5-126. Error Threshold Register (ERR_THRESHOLD) 531
5-127. Error Interrupt Flag Register (ERR_INTFLG) 532
5-128. Error Interrupt Flag Clear Register (ERR_INTCLR) 532
5-129. Data High Test Register (FDATAH_TEST) 533
5-130. Data Low Test Register (FDATAL_TEST) 533
5-131. ECC Test Address Register (FADDR_TEST) 533
5-132. ECC Test Register (FECC_TEST) 533

Copyright © 2012–2019, Texas Instruments Incorporated

5-133. ECC Control Register (FECC_CTRL) 534
5-134. Test Data Out High Register (FECC_FOUTH_TEST) 534
5-135. Test Data Out Low Register (FECC_FOUTL_TEST) 534
5-136. ECC Status Register (FECC_STATUS) 535
6-1. Device Boot Flow 540
6-2. M-Boot ROM Memory Map 541
6-3. M-Boot ROM Flow Diagram 550
6-4. M-Boot ROM Boot Status 552
6-5. Overview of Parallel GPIO Bootloader Operation 564
6-6. Parallel GPIO Bootloader Handshake Protocol 565
6-7. Parallel GPIO Mode Overview 566
6-8. Parallel GPIO Mode - Host Transfer Flow 567
6-9. 8-Bit Parallel GetWord Function 568
6-10. C-Boot ROM Memory Map 571
6-11. C-Boot ROM Vector Table Map 575
6-12. C-Boot ROM Flow Chart 583
6-13. Master Subsystem Application Procedure TO Send IPC TO C-Boot ROM 585
6-14. C-Boot ROM Handling on MTOCIPC 587
6-15. C-Boot ROM Health Status 591
6-16. Bootloader Basic Transfer Procedure 600
6-17. Overview of CopyData Function 601
6-18. Overview of SCI Bootloader Operation 601
6-19. Overview of SCI_Boot Function 602
6-20. Overview of SCI_GetWordData Function 603
6-21. SPI Loader 604
6-22. Data Transfer From EEPROM Flow 606
6-23. Overview of SPIA_GetWordData Function 606
6-24. EEPROM Device at Address 0x50 607
6-25. Overview of I2C_Boot Function 608
6-26. Random Read 609
6-27. Sequential Read 609
6-28. Overview of Parallel GPIO Bootloader Operation 610
6-29. Parallel GPIO Bootloader Handshake Protocol 611
6-30. Parallel GPIO Mode Overview 612
6-31. Parallel GPIO Mode - Host Transfer Flow 613
6-32. 8-Bit Parallel GetWord Function 614
6-33. Master Subsystem Application Flow to Start C-Boot ROM Loaders 615
6-34. Build a Binary Image for Bootload Using M-BOOT ROM 619
6-35. LM FLASH Programmer Configuration Screen 620
6-36. LM FLASH Programmer Interface Selection Screen 621
6-37. LM FLASH Programmer Serial Interface Configuration Screen 622
6-38. LM FLASH Programmer Binary Image Selection Screen 623
6-39. LM FLASH Programmer EMAC Interface Selection Screen 624
6-40. FLASH Programmer EMAC Bootload Binary Image Selection Screen 625
7-1. Multiple ePWM Modules 633
7-2. Submodules and Signal Connections for an ePWM Module 634
7-3. ePWM Submodules and Critical Internal Signal Interconnects 635
7-4. Time-Base Submodule 643
7-5. Time-Base Submodule Signals and Registers 645

Copyright © 2012–2019, Texas Instruments Incorporated

7-6. Time-Base Frequency and Period 647
7-7. Time-Base Counter Synchronization Scheme 4 648
7-8. Time-Base Up-Count Mode Waveforms 651
7-9. Time-Base Down-Count Mode Waveforms 652
7-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event 652
7-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event 653
7-12. Counter-Compare Submodule 653
7-13. Detailed View of the Counter-Compare Submodule 655
7-14. Counter-Compare Event Waveforms in Up-Count Mode 658
7-15. Counter-Compare Events in Down-Count Mode 659
7-16. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event 660
7-17. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event 660
7-18. Action-Qualifier Submodule 661
7-19. Action-Qualifier Submodule Inputs and Outputs 662
7-20. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs 663
7-21. AQCTLR[SHDWAQAMODE] 666
7-22. AQCTLR[SHDWAQBMODE] 666
7-23. Up-Down-Count Mode Symmetrical Waveform 668
7-24. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High 669
7-25. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low 670
7-26. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA 671
7-27. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low 673
7-28. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary 674
7-29. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low 675
7-30. Dead_Band Submodule 676
7-31. Configuration Options for the Dead-Band Submodule 679
7-32. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%) 681
7-33. PWM-Chopper Submodule 683
7-34. PWM-Chopper Submodule Operational Details 684
7-35. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only 684
7-36. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses 685
7-37. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses 686
7-38. Trip-Zone Submodule 687
7-39. Trip-Zone Submodule Mode Control Logic 691
7-40. Trip-Zone Submodule Interrupt Logic 692
7-41. Event-Trigger Submodule 693
7-42. Event-Trigger Submodule Inter-Connectivity of ADC Start of Conversion 694
7-43. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs 695
7-44. Event-Trigger Interrupt Generator 697
7-45. Event-Trigger SOCA Pulse Generator 698
7-46. Event-Trigger SOCB Pulse Generator 698
7-47. Digital-Compare Submodule High-Level Block Diagram 699
7-48. GPIO MUX-to-Trip Input Connectivity 700

Copyright © 2012–2019, Texas Instruments Incorporated

7-49. DCAEVT1 Event Triggering 703
7-50. DCAEVT2 Event Triggering 703
7-51. DCBEVT1 Event Triggering 704
7-52. DCBEVT2 Event Triggering 704
7-53. Event Filtering 705
7-54. Blanking Window Timing Diagram 706
7-55. Simplified ePWM Module 707
7-56. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave 708
7-57. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$ 709
7-58. Buck Waveforms for (Note: Only three bucks shown here) 710
7-59. Control of Four Buck Stages. (Note: $F_{PWM2} = N \times F_{PWM1}$) 712
7-60. Buck Waveforms for (Note: $F_{PWM2} = F_{PWM1)}$) 713
7-61. Control of Two Half-H Bridge Stages ($F_{PWM2} = N \times F_{PWM1}$) 715
7-62. Half-H Bridge Waveforms for (Note: Here $F_{PWM2} = F_{PWM1}$) 716
7-63. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control 718
7-64. 3-Phase Inverter Waveforms for (Only One Inverter Shown) 719
7-65. Configuring Two PWM Modules for Phase Control 721
7-66. Timing Waveforms Associated With Phase Control Between 2 Modules 722
7-67. Control of a 3-Phase Interleaved DC/DC Converter 723
7-68. 3-Phase Interleaved DC/DC Converter Waveforms for 724
7-69. Controlling a Full-H Bridge Stage ($F_{PWM2} = F_{PWM1)}$ 726
7-70. ZVS Full-H Bridge Waveforms 727
7-71. Peak Current Mode Control of a Buck Converter 729
7-72. Peak Current Mode Control Waveforms for 729
7-73. Control of Two Resonant Converter Stages 731
7-74. H-Bridge LLC Resonant Converter PWM Waveforms 731
7-75. Time-Base Period and Mirror 2 Register (TBPRD / TBPRDM2) 733
7-76. Time-Base Period High-Resolution and Mirror 2 Register (TBPRDHR / TBPRDHRM2) 733
7-77. Time-Base Period Mirror Register (TBPRDM) 733
7-78. Time-Base Period High-Resolution Mirror Register (TBPRDHRM) 734
7-79. Time-Base Phase Register and Mirror Register (TBPHS / TBPHSM) 734
7-80. Time-Base Phase High-Resolution Register and Mirror Register (TBPHSHR / TBPHSHRM) 735
7-81. Time-Base Counter Register (TBCTR) 735
7-82. Time-Base Control Register (TBCTL) 735
7-83. Time-Base Status Register (TBSTS) 738
7-84. High-Resolution Period Control Register (HRPCTL) 738
7-85. Time-Base Control Register 2 (TBCTL2) 739
7-86. EPWMx Link Register (EPWMXLINK) 739
7-87. Counter-Compare Control Register (CMPCTL) 742
7-88. Compare Control Register (CMPCTL2) 743
7-89. Compare A High-Resolution and Mirror 2 Register (CMPAHR / CMPAHRM2) 744
7-90. Counter-Compare A and Mirror 2 Register (CMPA / CMPAM2) 745
7-91. Compare A High-Resolution Mirror Register (CMPAHRM) 745
7-92. Counter-Compare A Mirror Register (CMPAM) 745
7-93. Counter-Compare B Register (CMPBM) 746
7-94. Counter-Compare B Register (CMPB) 746
7-95. Counter-Compare C Register (CMPC) 747
7-96. Counter-Compare D Register (CMPD) 747
7-97. Compare B High-Resolution Register (CMPBHR) 748

Copyright © 2012–2019, Texas Instruments Incorporated

7-98. Compare B High-Resolution Mirror Register (CMPBHRM) 748
7-99. Action-Qualifier Output A Control Register and Mirror Register (AQCTLA / AQCTLAM) 749
7-100. Action-Qualifier Output B Control Register and Mirror Register (AQCTLB / AQCTLBM) 750
7-101. Action-Qualifier Software Force Register and Mirror Register (AQSFRC / AQSFRCM) 751
7-102. Action-Qualifier Continuous Software Force Register and Mirror Register (AQCSFRC / AQCSFRCM) 752
7-103. Action Qualifier Control Register (AQCTLR) 753
7-104. Dead-Band Generator Control Register (DBCTL) 754
7-105. Dead-Band Generator Rising Edge Delay and Mirror Register (DBRED / DBREDM) 756
7-106. Dead-Band Generator Falling Edge Delay and Mirror Register (DBFED / DBFEDM) 756
7-107. Dead Band Rising Edge Delay High-Resolution Register (DBREDHR) 756
7-108. Dead Band Falling Edge Delay High-Resolution Register (DBFEDHR) 756
7-109. PWM-Chopper Control Register (PCCTL) 758
7-110. Trip-Zone Select Register (TZSEL) 760
7-111. Trip-Zone Control Register (TZCTL) 761
7-112. Trip-Zone Enable Interrupt Register (TZEINT) 762
7-113. Trip-Zone Flag Register (TZFLG) 763
7-114. Trip-Zone Clear Register and Mirror Register (TZCLR / TZCLRM) 764
7-115. Trip-Zone Force Register (TZFRC) 765
7-116. Trip-Zone Digital Compare Event Select Register (TZDCSEL) 765
7-117. Digital Compare Trip Select (DCTRIPSEL) 767
7-118. Digital Compare A Control Register (DCACTL) 768
7-119. Digital Compare B Control Register (DCBCTL) 769
7-120. Digital Compare Filter Control Register (DCFCTL) 769
7-121. Digital Compare Capture Control Register (DCCAPCTL) 770
7-122. Digital Compare Counter Capture Register (DCCAP) 770
7-123. Digital Compare Filter Offset Register (DCFOFFSET) 771
7-124. Digital Compare Filter Offset Counter Register (DCFOFFSETCNT) 771
7-125. Digital Compare Filter Window Register (DCFWINDOW) 772
7-126. Digital Compare Filter Window Counter Register (DCFWINDOWCNT) 772
7-127. Digital Compare A High Trip Input Select (DCAHTRIPSEL) (EALLOW-protected) 772
7-128. Digital Compare A Low Trip Input Select (DCALTRIPSEL) (EALLOW-protected) 774
7-129. Digital Compare B High Trip Input Select (DCBHTRIPSEL) (EALLOW-protected) 775
7-130. Digital Compare B Low Trip Input Select (DCBLTRIPSEL) (EALLOW-protected) 776
7-131. GPIO Trip Input Select Register (GPTRIPxSEL) 778
7-132. Event-Trigger Selection Register (ETSEL) 780
7-133. Event-Trigger Prescale Register (ETPS) 781
7-134. Event-Trigger Interrupt Pre-Scale Register (ETINTPS) 783
7-135. Event-Trigger SOC Pre-Scale Register (ETSOCPS) 784
7-136. Event-Trigger Flag Register (ETFLG) 785
7-137. Event-Trigger Clear Register and Mirror Register (ETCLR / ETCLRM) 786
7-138. Event-Trigger Force Register (ETFRC) 786
7-139. Event-Trigger Counter Initialization Control Register (ETCNTINITCTL) 787
7-140. Event-Trigger Counter Initialization Register (ETCNTINIT) 787
8-1. Capture and APWM Modes of Operation 792
8-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode 793
8-3. Capture Function Diagram 794
8-4. Event Prescale Control 795
8-5. Prescale Function Waveforms 795
8-6. Details of the Continuous/One-shot Block 796

Copyright © 2012–2019, Texas Instruments Incorporated

8-7. Details of the Counter and Synchronization Block 797
8-8. Interrupts in eCAP Module 798
8-9. PWM Waveform Details Of APWM Mode Operation 799
8-10. Time-Base Frequency and Period Calculation 800
8-11. Time-Stamp Counter Register (TSCTR) 801
8-12. Counter Phase Control Register (CTRPHS) 801
8-13. Capture-1 Register (CAP1) 801
8-14. Capture-2 Register (CAP2) 801
8-15. Capture-3 Register (CAP3) 802
8-16. Capture-4 Register (CAP4) 802
8-17. ECAP Control Register 1 (ECCTL1) 802
8-18. ECAP Control Register 2 (ECCTL2) 803
8-19. ECAP Interrupt Enable Register (ECEINT) 806
8-20. ECAP Interrupt Flag Register (ECFLG) 807
8-21. ECAP Interrupt Clear Register (ECCLR) 807
8-22. ECAP Interrupt Forcing Register (ECFRC) 808
8-23. Capture Sequence for Absolute Time-stamp and Rising Edge Detect 810
8-24. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect 812
8-25. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect 814
8-26. Capture Sequence for Delta Mode Time-stamp With Rising and Falling Edge Detect 816
8-27. PWM Waveform Details of APWM Mode Operation 818
9-1. Optical Encoder Disk 821
9-2. QEP Encoder Output Signal for Forward/Reverse Movement 821
9-3. Index Pulse Example 822
9-4. Functional Block Diagram of the eQEP Peripheral 824
9-5. Functional Block Diagram of Decoder Unit 826
9-6. Quadrature Decoder State Machine 827
9-7. Quadrature-clock and Direction Decoding 828
9-8. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or 0xF9F) 830
9-9. Position Counter Underflow/Overflow (QPOSMAX = 4) 831
9-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1) 832
9-11. Strobe Event Latch (QEPCTL[SEL] = 1) 833
9-12. eQEP Position-compare Unit 834
9-13. eQEP Position-compare Event Generation Points 835
9-14. eQEP Position-compare Sync Output Pulse Stretcher 835
9-15. eQEP Edge Capture Unit 837
9-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010) 837
9-17. eQEP Edge Capture Unit - Timing Details 838
9-18. eQEP Watchdog Timer 839
9-19. eQEP Unit Time Base 839
9-20. EQEP Interrupt Generation 840
9-21. QEP Decoder Control (QDECCTL) Register 840
9-22. eQEP Control (QEPCTL) Register 841
9-23. eQEP Position-compare Control (QPOSCTL) Register 843
9-24. eQEP Capture Control (QCAPCTL) Register 844
9-25. eQEP Position Counter (QPOSCNT) Register 844
9-26. eQEP Position Counter Initialization (QPOSINIT) Register 844
9-27. eQEP Maximum Position Count Register (QPOSMAX) Register 845
9-28. eQEP Position-compare (QPOSCMP) Register 845

Copyright © 2012–2019, Texas Instruments Incorporated

9-29. eQEP Index Position Latch (QPOSILAT) Register 845
9-30. eQEP Strobe Position Latch (QPOSSLAT) Register 845
9-31. eQEP Position Counter Latch (QPOSLAT) Register 846
9-32. eQEP Unit Timer (QUTMR) Register 846
9-33. eQEP Register Unit Period (QUPRD) Register 846
9-34. eQEP Watchdog Timer (QWDTMR) Register 846
9-35. eQEP Watchdog Period (QWDPRD) Register 847
9-36. eQEP Interrupt Enable (QEINT) Register 847
9-37. eQEP Interrupt Flag (QFLG) Register 848
9-38. eQEP Interrupt Clear (QCLR) Register 849
9-39. eQEP Interrupt Force (QFRC) Register 850
9-40. eQEP Status (QEPSTS) Register 851
9-41. eQEP Capture Timer (QCTMR) Register 852
9-42. eQEP Capture Period (QCPRD) Register 852
9-43. eQEP Capture Timer Latch (QCTMRLAT) Register 852
9-44. eQEP Capture Period Latch (QCPRDLAT) Register 853
10-1. Analog Subsystem Block Diagram 855
10-2. Simplified ACIB Model 856
10-3. 16-bit Write 857
10-4. 32-bit Write 857
10-5. 16-bit Read 857
10-6. 32-bit Read 858
10-7. 64-bit Read 858
10-8. ADC Trigger 858
10-9. ADC Interrupt 859
10-10. ADC Module Block Diagram 860
10-11. SOC Block Diagram 861
10-12. ADCINx Input Model 863
10-13. ONESHOT Single Conversion 864
10-14. Round Robin Priority Example 866
10-15. High Priority Example 867
10-16. Interrupt Structure 869
10-17. ADC Control Register 1 (ADCCTL1) (Address Offset 00h) 873
10-18. ADC Control Register 2 (ADCCTL2) (Address Offset 01h) 875
10-19. ADC Interrupt Flag Register (ADCINTFLG) (Address Offset 04h) 875
10-20. ADC Interrupt Flag Clear Register (ADCINTFLGCLR) (Address Offset 05h) 876
10-21. ADC Interrupt Overflow Register (ADCINTOVF) (Address Offset 06h) 876
10-22. ADC Interrupt Overflow Clear Register (ADCINTOVFCLR) (Address Offset 07h) 876
10-23. Interrupt Select 1 And 2 Register (INTSEL1N2) (Address Offset 08h) 877
10-24. Interrupt Select 3 And 4 Register (INTSEL3N4) (Address Offset 09h) 877
10-25. Interrupt Select 5 And 6 Register (INTSEL5N6) (Address Offset 0Ah) 877
10-26. Interrupt Select 7 And 8 Register (INTSEL7N8) (Address Offset 0Bh) 877
10-27. Interrupt Select 9 And 10 Register (INTSEL9N10) (Address Offset 0Ch) 878
10-28. ADC Start of Conversion Priority Control Register (SOCPRICTL) 879
10-29. ADC Sample Mode Register (ADCSAMPLEMODE) (Address Offset 12h) 881
10-30. ADC Interrupt Trigger SOC Select 1 Register (ADCINTSOCSEL1) (Address Offset 14h) 882
10-31. ADC Interrupt Trigger SOC Select 2 Register (ADCINTSOCSEL2) (Address Offset 15h) 883
10-32. ADC SOC Flag 1 Register (ADCSOCFLG1) (Address Offset 18h) 883
10-33. ADC SOC Force 1 Register (ADCSOCFRC1) (Address Offset 1Ah) 883

Copyright © 2012–2019, Texas Instruments Incorporated

10-34. ADC SOC Overflow 1 Register (ADCSOCOVF1) (Address Offset 1Ch) 884
10-35. ADC SOC Overflow Clear 1 Register (ADCSOCOVFCLR1) (Address Offset 1Eh) 884
10-36. ADC SOC0 - SOC15 Control Registers (ADCSOCxCTL) (Address Offset 20h - 2Fh) 885
10-37. ADC Reference/Gain Trim Register (ADCREFTRIM) (Address Offset 40h) 887
10-38. ADC Offset Trim Register (ADCOFFTRIM) (Address Offset 41h) 887
10-39. ADC Revision Register (ADCREV) (Address Offset 4Fh) 888
10-40. ADC RESULT0 - RESULT15 Registers (ADCRESULTx) (PF1 Block Address Offset 00h - 0Fh) 888
10-41. ADC Interrupt Overflow Detect Register (INTOVF) 890
10-42. ADC Interrupt Overflow Clear Register (INTOVFCLR) 891
10-43. Control System: Lock Register (CLOCK) 892
10-44. Control System: ACIB Status Register (CCIBSTATUS) 893
10-45. Control System: Clock Control Register (CCLKCTL) 894
10-46. ADC Start of Conversion Trigger Overflow Detect Register (TRIGOVF) 895
10-47. ADC Start of Conversion Trigger Overflow Flag Clear Register (TRIGOVFCLR) 896
10-48. ADC Start of Conversion Trigx Input Select Register (TRIGxSEL) 897
10-49. Timing Example For Sequential Mode / Late Interrupt Pulse 898
10-50. Timing Example For Sequential Mode / Early Interrupt Pulse 899
10-51. Timing Example For Simultaneous Mode / Late Interrupt Pulse 900
10-52. Timing Example For Simultaneous Mode / Early Interrupt Pulse 901
10-53. Comparator Block Diagram 902
10-54. Comparator 902
10-55. Comparator Control (COMPCTL) Register 904
10-56. Compare Output Status (COMPSTS) Register 905
10-57. DAC Value (DACVAL) Register 905
10-58. DAC Test (DACTEST) Register 906
11-1. DMA Block Diagram 913
11-2. Peripheral Interrupt Trigger Input Diagram 914
11-3. 4-Stage Pipeline DMA Transfer 916
11-4. 4-Stage Pipeline With One Read Stall (McBSP as source) 916
11-5. Arbitration when Accessing ACIB 918
11-6. DMA State Diagram 923
11-7. Overrun Detection Logic 925
11-8. DMA Control Register (DMACTRL) 927
11-9. Debug Control Register (DEBUGCTRL) 929
11-10. Revision Register (REVISION) 929
11-11. Priority Control Register 1 (PRIORITYCTRL1) 930
11-12. Priority Status Register (PRIORITYSTAT) 931
11-13. Mode Register (MODE) 932
11-14. Control Register (CONTROL) 934
11-15. Burst Size Register (BURST_SIZE) 936
11-16. Burst Count Register (BURST_COUNT) 936
11-17. Source Burst Step Size Register (SRC_BURST_STEP) 937
11-18. Destination Burst Step Register Size (DST_BURST_STEP) 938
11-19. Transfer Size Register (TRANSFER_SIZE) 938
11-20. Transfer Count Register (TRANSFER_COUNT) 939
11-21. Source Transfer Step Size Register (SRC_TRANSFER_STEP) 939
11-22. Destination Transfer Step Size Register (DST_TRANSFER_STEP) 940
11-23. Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE) 940
11-24. Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT) 941

Copyright © 2012–2019, Texas Instruments Incorporated

11-25. Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP) 941
11-26. Shadow Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW)..... 942
11-27. Active Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR/DST_BEG_ADDR) 942
11-28. Shadow Destination Begin and Current Address Pointer Registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW) 943
11-29. Active Destination Begin and Current Address Pointer Registers (SRC_ADDR/DST_ADDR)..... 943
12-1. SPI CPU Interface 945
12-2. Serial Peripheral Interface Module Block Diagram 947
12-3. SPI Master/Slave Connection 950
12-4. SPICLK Signal Options 954
12-5. SPI: SPICLK-CLKOUT Characteristic When (BRR + 1) is Odd, BRR > 3, and CLOCK POLARITY = 1 954
12-6. Five Bits per Character..... 956
12-7. SPI FIFO Interrupt Flags and Enable Logic Generation..... 957
12-8. SPI 3-wire Master Mode 958
12-9. SPI 3-wire Slave Mode..... 959
12-10. SPI Digital Audio Receiver Configuration Using 2 SPIs 961
12-11. Standard Right-Justified Digital Audio Data Format..... 961
12-12. SSI and SPI Connections for Loopback Mode..... 962
12-13. SPI Configuration Control Register (SPICCR) — Address 7040h 964
12-14. SPI Operation Control Register (SPICTL) — Address 7041h 965
12-15. SPI Status Register (SPIST) — Address 7042h..... 966
12-16. SPI Baud Rate Register (SPIBRR) — Address 7044h 967
12-17. SPI Emulation Buffer Register (SPIRXEMU) — Address 7046h 968
12-18. SPI Serial Receive Buffer Register (SPIRXBUF) — Address 7047h 968
12-19. SPI Serial Transmit Buffer Register (SPITXBUF) — Address 7048h..... 969
12-20. SPI Serial Data Register (SPIDAT) — Address 7049h 969
12-21. SPI FIFO Transmit (SPIFFTX) Register − Address 704Ah 970
12-22. SPI FIFO Receive (SPIFFRX) Register − Address 704Bh..... 970
12-23. SPI FIFO Control (SPIFFCT) Register − Address 704Ch 971
12-24. SPI Priority Control Register (SPIPRI) — Address 704Fh 972
12-25. CLOCK POLARITY = 0, CLOCK PHASE = 0 (All data transitions are during the rising edge, non-delayed clock. Inactive level is low.)..... 973
12-26. CLOCK POLARITY = 0, CLOCK PHASE = 1 (All data transitions are during the rising edge, but delayed by half clock cycle. Inactive level is low.) 974
12-27. CLOCK POLARITY = 1, CLOCK PHASE = 0 (All data transitions are during the falling edge. Inactive level is high.) 975
12-28. CLOCK POLARITY = 1, CLOCK PHASE = 1 (All data transitions are during the falling edge, but delayed by half clock cycle. Inactive level is high.)..... 976
12-29. $\overline{\text{SPISTE}}$ Behavior in Master Mode (Master lowers $\overline{\text{SPISTE}}$ during the entire 16 bits of transmission.)..... 977
12-30. $\overline{\text{SPISTE}}$ Behavior in Slave Mode (Slave's $\overline{\text{SPISTE}}$ is lowered during the entire 16 bits of transmission.) ... 978
13-1. SCI CPU Interface 980
13-2. Serial Communications Interface (SCI) Module Block Diagram 981
13-3. Typical SCI Data Frame Formats..... 984
13-4. Idle-Line Multiprocessor Communication Format..... 985
13-5. Double-Buffered WUT and TXSHF..... 986
13-6. Address-Bit Multiprocessor Communication Format 987
13-7. SCI Asynchronous Communications Format..... 988
13-8. SCI RX Signals in Communication Modes 988
13-9. SCI TX Signals in Communications Mode 989

Copyright © 2012–2019, Texas Instruments Incorporated

13-10. SCI FIFO Interrupt Flags and Enable Logic 991
13-11. UART and SCI Connections for Loopback Mode 993
13-12. SCI Communication Control Register (SCICCR) — Address 7050h 995
13-13. SCI Control Register 1 (SCICTL1) — Address 7051h 996
13-14. Baud-Select MSbyte Register (SCIHBAUD) — Address 7052h 998
13-15. Baud-Select LSbyte Register (SCILBAUD) — Address 7053h 998
13-16. SCI Control Register 2 (SCICTL2) — Address 7054h 999
13-17. SCI Receiver Status Register (SCIRXST) — Address 7055h 999
13-18. Register SCIRXST Bit Associations — Address 7055h 1001
13-19. Emulation Data Buffer Register (SCIRXEMU) — Address 7056h 1001
13-20. SCI Receive Data Buffer Register (SCIRXBUF) — Address 7057h 1001
13-21. Transmit Data Buffer Register (SCITXBUF) — Address 7059h 1002
13-22. SCI FIFO Transmit (SCIFFTX) Register — Address 705Ah 1002
13-23. SCI FIFO Receive (SCIFFRX) Register — Address 705Bh 1003
13-24. SCI FIFO Control (SCIFFCT) Register — Address 705Ch 1004
13-25. SCI Priority Control Register (SCIPRI) — Address 705Fh 1006
14-1. Multiple I2C Modules Connected 1008
14-2. I2C Module Conceptual Block Diagram 1010
14-3. Clocking Diagram for the I2C Module 1011
14-4. Bit Transfer on the I2C-Bus 1012
14-5. I2C Module START and STOP Conditions 1013
14-6. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown) 1014
14-7. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR) 1014
14-8. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR) 1014
14-9. I2C Module Free Data Format (FDF = 1 in I2CMDR) 1014
14-10. Repeated START Condition (in This Case, 7-Bit Addressing Format) 1015
14-11. Synchronization of Two I2C Clock Generators During Arbitration 1016
14-12. Arbitration Procedure Between Two Master-Transmitters 1017
14-13. Enable Paths of the I2C Interrupt Requests 1018
14-14. I2C Mode Register (I2CMDR) 1020
14-15. Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit 1023
14-16. I2C Extended Mode Register (I2CEMDR) 1023
14-17. BCM Bit, Slave Transmitter Mode 1024
14-18. I2C Interrupt Enable Register (I2CIER) 1025
14-19. I2C Status Register (I2CSTR) 1026
14-20. I2C Interrupt Source Register (I2CISRC) 1028
14-21. I2C Prescaler Register (I2CPSC) 1029
14-22. The Roles of the Clock Divide-Down Values (ICCL and ICCH) 1030
14-23. I2C Clock Low-Time Divider Register (I2CCLKL) 1030
14-24. I2C Clock High-Time Divider Register (I2CCLKH) 1030
14-25. I2C Slave Address Register (I2CSAR) 1031
14-26. I2C Own Address Register (I2COAR) 1031
14-27. I2C Data Count Register (I2CCNT) 1032
14-28. I2C Data Receive Register (I2CDRR) 1032
14-29. I2C Data Transmit Register (I2CDXR) 1033
14-30. I2C Transmit FIFO Register (I2CFFTX) 1033
14-31. I2C Receive FIFO Register (I2CFFRX) 1034
15-1. Conceptual Block Diagram of the McBSP 1039
15-2. McBSP Data Transfer Paths 1040

Copyright © 2012–2019, Texas Instruments Incorporated

15-3. Companding Processes 1041
15-4. μ-Law Transmit Data Companding Format 1041
15-5. A-Law Transmit Data Companding Format 1041
15-6. Two Methods by Which the McBSP Can Compand Internal Data 1042
15-7. Example - Clock Signal Control of Bit Transfer Timing 1042
15-8. McBSP Operating at Maximum Packet Frequency 1044
15-9. Single-Phase Frame for a McBSP Data Transfer 1045
15-10. Dual-Phase Frame for a McBSP Data Transfer 1045
15-11. Implementing the AC97 Standard With a Dual-Phase Frame 1046
15-12. Timing of an AC97-Standard Data Transfer Near Frame Synchronization 1046
15-13. McBSP Reception Physical Data Path 1047
15-14. McBSP Reception Signal Activity 1047
15-15. McBSP Transmission Physical Data Path 1048
15-16. McBSP Transmission Signal Activity 1048
15-17. Conceptual Block Diagram of the Sample Rate Generator 1050
15-18. Possible Inputs to the Sample Rate Generator and the Polarity Bits 1052
15-19. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1 1054
15-20. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3 1055
15-21. Overrun in the McBSP Receiver 1057
15-22. Overrun Prevented in the McBSP Receiver 1058
15-23. Possible Responses to Receive Frame-Synchronization Pulses 1058
15-24. An Unexpected Frame-Synchronization Pulse During a McBSP Reception 1059
15-25. Proper Positioning of Frame-Synchronization Pulses 1060
15-26. Data in the McBSP Transmitter Overwritten and Thus Not Transmitted 1060
15-27. Underflow During McBSP Transmission 1061
15-28. Underflow Prevented in the McBSP Transmitter 1062
15-29. Possible Responses to Transmit Frame-Synchronization Pulses 1062
15-30. An Unexpected Frame-Synchronization Pulse During a McBSP Transmission 1063
15-31. Proper Positioning of Frame-Synchronization Pulses 1064
15-32. Alternating Between the Channels of Partition A and the Channels of Partition B 1066
15-33. Reassigning Channel Blocks Throughout a McBSP Data Transfer 1067
15-34. McBSP Data Transfer in the 8-Partition Mode 1068
15-35. Activity on McBSP Pins for the Possible Values of XMCM 1071
15-36. Typical SPI Interface 1072
15-37. SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 0, and CLKRP = 0 1074
15-38. SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 0, CLKRP = 1 1074
15-39. SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 1, and CLKRP = 0 1074
15-40. SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 1, CLKRP = 1 1074
15-41. SPI Interface with McBSP Used as Master 1076
15-42. SPI Interface With McBSP Used as Slave 1077
15-43. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 0 1084
15-44. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 1 1084
15-45. Companding Processes for Reception and for Transmission 1085
15-46. Range of Programmable Data Delay 1086
15-47. 2-Bit Data Delay Used to Skip a Framing Bit 1087
15-48. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge 1091
15-49. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods 1092
15-50. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge 1094
15-51. Unexpected Frame-Synchronization Pulse With (R/X) FIG = 0 1104

Copyright © 2012–2019, Texas Instruments Incorporated

15-52. Unexpected Frame-Synchronization Pulse With (R/X) FIG = 1 1105
15-53. Companding Processes for Reception and for Transmission 1105
15-54. μ-Law Transmit Data Companding Format 1106
15-55. A-Law Transmit Data Companding Format 1106
15-56. Range of Programmable Data Delay 1107
15-57. 2-Bit Data Delay Used to Skip a Framing Bit 1107
15-58. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge .. 1111
15-59. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods 1111
15-60. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge .. 1113
15-61. Four 8-Bit Data Words Transferred To/From the McBSP 1117
15-62. One 32-Bit Data Word Transferred To/From the McBSP 1117
15-63. 8-Bit Data Words Transferred at Maximum Packet Frequency 1118
15-64. Configuring the Data Stream of as a Continuous 32-Bit Word 1118
15-65. Data Receive Registers (DRR2 and DRR1) 1120
15-66. Data Transmit Registers (DXR2 and DXR1) 1120
15-67. Serial Port Control 1 Register (SPCR1) 1121
15-68. Serial Port Control 2 Register (SPCR2) 1124
15-69. Receive Control Register 1 (RCR1) 1126
15-70. Receive Control Register 2 (RCR2) 1127
15-71. Transmit Control 1 Register (XCR1) 1129
15-72. Transmit Control 2 Register (XCR2) 1130
15-73. Sample Rate Generator 1 Register (SRGR1) 1132
15-74. Sample Rate Generator 2 Register (SRGR2) 1132
15-75. Multichannel Control 1 Register (MCR1) 1134
15-76. Multichannel Control 2 Register (MCR2) 1136
15-77. Pin Control Register (PCR) 1138
15-78. Receive Channel Enable Registers (RCERA...RCERH) 1140
15-79. Transmit Channel Enable Registers (XCERA...XCERH) 1142
15-80. Receive Interrupt Generation 1144
15-81. Transmit Interrupt Generation 1145
15-82. McBSP Interrupt Enable Register (MFFINT) 1146
16-1. μDMA Block Diagram 1150
16-2. Example of Ping-Pong μDMA Transaction 1156
16-3. Memory Scatter-Gather, Setup and Configuration 1158
16-4. Memory Scatter-Gather, μDMA Copy Sequence 1159
16-5. Peripheral Scatter-Gather, Setup and Configuration 1161
16-6. Peripheral Scatter-Gather, μDMA Copy Sequence 1162
16-7. DMA Channel Source Address End Pointer (DMASRCENDP) Register 1171
16-8. DMA Channel Destination Address End Pointer (DMADSTENDP) Register 1171
16-9. DMA Channel Control Word (DMACHCTL) Register 1172
16-10. DMA Status (DMASTAT) Register 1175
16-11. DMA Configuration (DMACFG) Register 1176
16-12. DMA Channel Control Base Pointer (DMACTLBASE) Register 1176
16-13. DMA Alternate Channel Control Base Pointer (DMAALTBASE) Register 1177
16-14. DMA Channel Wait-on-Request Status (DMAWAITSTAT) Register 1177
16-15. DMA Channel Software Request (DMASWREQ) Register 1178
16-16. DMA Channel Useburst Set (DMAUSEBURSTSET) Register 1178
16-17. DMA Channel Useburst Clear (DMAUSEBURSTCLR) Register 1178
16-18. DMA Channel Request Mask Set (DMAREQMASKSET) Register 1179

Copyright © 2012–2019, Texas Instruments Incorporated

16-19. DMA Channel Request Mask Clear (DMAREQMASKCLR) Register 1179
16-20. DMA Channel Enable Set (DMAENASET) Register 1180
16-21. DMA Channel Enable Clear (DMAENACLR) Register 1180
16-22. DMA Channel Primary Alternate Set (DMAALTSET) Register 1180
16-23. DMA Channel Primary Alternate Clear (DMAALTCLR) Register 1181
16-24. DMA Channel Priority Set (DMAPRIOSET) Register 1181
16-25. DMA Channel Priority Clear (DMAPRIOCLR) Register 1181
16-26. DMA Bus Error Clear (DMAERRCLR) Register 1182
16-27. DMA Channel Assignment (DMACHALT) Register 1182
16-28. DMA Channel Map Assignment (DMACHMAP0) Register 1183
16-29. DMA Channel Map Assignment (DMACHMAP1) Register 1184
16-30. DMA Channel Map Assignment (DMACHMAP2) Register 1185
16-31. DMA Channel Map Assignment (DMACHMAP3) Register 1186
16-32. DMA Peripheral Identification 0 (DMAPeriphID0) Register 1187
16-33. DMA Peripheral Identification 1 (DMAPeriphID1) Register 1187
16-34. DMA Peripheral Identification 2 (DMAPeriphID2) Register 1187
16-35. DMA Peripheral Identification 3 (DMAPeriphID3) Register 1187
16-36. DMA Peripheral Identification 4 (DMAPeriphID4) Register 1188
16-37. DMA PrimeCell Identification 0 (DMAPCellID0) Register 1188
16-38. DMA PrimeCell Identification 1 (DMAPCellID1) Register 1188
16-39. DMA PrimeCell Identification 2 (DMAPCellID2) Register 1189
16-40. DMA PrimeCell Identification 3 (DMAPCellID3) Register 1189
17-1. EPI Block Diagram 1192
17-2. SDRAM Non-Blocking Read Cycle 1197
17-3. SDRAM Normal Read Cycle 1198
17-4. SDRAM Write Cycle 1199
17-5. iRDY Access Stalls 1209
17-6. iRDY Signal Connection 1209
17-7. Example Schematic for Muxed Host-Bus 16 Mode 1210
17-8. Host-Bus Read Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0, ALEHIGH = 1 1212
17-9. Host-Bus Write Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0, ALEHIGH = 1 1213
17-10. Host-Bus Write Cycle with Multiplexed Address and Data, MODE = 0x0, WRHIGH = 0, RDHIGH = 0, ALEHIGH = 1 1213
17-11. Host-Bus Write Cycle with Multiplexed Address and Data and ALE with Dual or Quad $\overline{CS}$ 1214
17-12. Continuous Read Mode Accesses 1214
17-13. Write Followed by Read to External FIFO 1214
17-14. Two-Entry FIFO 1215
17-15. Single-Cycle Write Access, FRM50 = 0, FRMCNT = 0, WR2CYC = 0 1219
17-16. Two-Cycle Read, Write Accesses, FRM50 = 0, FRMCNT = 0, RD2CYC = 1, WR2CYC = 1 1219
17-17. Read Accesses, FRM50 = 0, FRMCNT = 0, RD2CYC = 1 1219
17-18. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 0 1220
17-19. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 1 1220
17-20. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 2 1220
17-21. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 0 1221
17-22. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 1 1221
17-23. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 2 1221
17-24. iRDY Signal Operation, FRM50 = 0, FRMCNT = 0, and RD2CYC = 1 1222
17-25. EPI Clock Operation, CLKGATE = 1, WR2CYC = 0 1222
17-26. EPI Clock Operation, CLKGATE = 1, WR2CYC = 1 1223

Copyright © 2012–2019, Texas Instruments Incorporated

17-27. C28x Master and Control Subsystem Access to EPI 1223
17-28. EPI Configuration Register (EPICFG) [offset 0x000] 1228
17-29. EPI Main Baud Rate (EPIBAUD) Register [offset 0x004] 1229
17-30. EPI Main Baud Rate (EPIBAUD2) Register [offset 0x008] 1230
17-31. EPI SDRAM Configuration (EPISDRAMCFG) Register [offset 0x010] 1231
17-32. EPI Host-Bus 8 Configuration (EPIHB8CFG) Register [offset 0x010] 1232
17-33. EPI Host-Bus 16 Configuration (EPIHB16CFG) Register [offset 0x010] 1235
17-34. EPI General-Purpose Configuration (EPIGPCFG) Register [offset 0x010] 1238
17-35. EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2) Register [offset 0x014] 1241
17-36. EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2) Register [offset 0x014] 1244
17-37. EPI General-Purpose Configuration 2 (EPIGPCFG2)Register [offset 0x014] 1247
17-38. EPI Address Map (EPIADDRMAP) Register [offset 0x01C] 1248
17-39. EPI Read Size 0 (EPIRSIZE0) Register [offset 0x020] and EPI Read Size 1 (EPIRSIZE1) Register [offset 0x030] 1250
17-40. EPI Read Address 0 (EPIRADDR0) Register [offset 0x024] and EPI Read Address 1 (EPIRADDR1) Register [offset 0x034] 1251
17-41. EPI Non-Blocking Read Data 0 (EPIRPSTD0) Register [offset 0x028] and EPI Non-Blocking Read Data 1 (EPIRPSTD1) Register [offset 0x038] 1252
17-42. EPI Status (EPISTAT) Register [offset 0x060] 1253
17-43. EPI Read FIFO Count (EPIRFIFOCNT) Register [offset 0x06C] 1254
17-44. EPI Read FIFO (EPIREADFIFO) Register [offset 0x070] and EPI Read FIFO Alias 1-7 (EPIREADFIFO1-7) Registers [offset 0x074 - 0x08C] 1255
17-45. EPI FIFO Level Selects (EPIFIFOLVL) Register [offset 0x200] 1255
17-46. EPI Write FIFO Count (EPIWFIFOCNT) Register [offset 0x204] 1257
17-47. EPI DMA Transmit Count (EPIDMATXCNT) Register [offset 0x208] 1258
17-48. EPI Interrupt Mask (EPIIM) Register [offset 0x210] 1258
17-49. EPI Raw Interrupt Status (EPIRIS) Register [offset 0x214] 1259
17-50. EPI Masked Interrupt Status (EPIMIS) Register [offset 0x218] 1260
17-51. EPI Error Interrupt Status and Clear (EPIEISC) Register [offset 0x21C] 1262
17-52. EPI Host-Bus 8 Configuration 3 Register (EPIHB8CFG3) [offset 0x308] 1263
17-53. EPI Host-Bus 16 Configuration 3 Register (EPIHB16CFG3) [offset 0x308] 1264
17-54. EPI Host-Bus 8 Configuration 4 Register (EPIHB8CFG4) [offset 0x30C] 1265
17-55. EPI Host-Bus 16 Configuration 4 Register (EPIHB16CFG4) [offset 0x30C] 1267
17-56. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME) [offset 0x310] 1268
17-57. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME) [offset 0x310] 1269
17-58. EPI Host-Bus 8 Timing Extension (EPIHB8TIME2) Register [offset 0x314] 1270
17-59. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME2) [offset 0x314] 1271
17-60. EPI Host-Bus 8 Timing Extension (EPIHB8TIME3) Register [offset 0x318] 1272
17-61. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME3) [offset 0x318] 1273
17-62. EPI Host-Bus 8 Timing Extension (EPIHB8TIME4) Register [offset 0x31C] 1274
17-63. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME4) [offset 0x31C] 1275
17-64. CEPIRTWCFG Register 1276
17-65. CEPIRTWCNT Register 1276
17-66. CEPIRTWPRD Register 1277
17-67. CEPISTATUS Register 1277
17-68. MEMPROT Register 1277
18-1. USB Block Diagram 1280
18-2. Function Address Register (USBFADDR) 1300
18-3. Power Management Register (USBPOWER) in OTG A/Host Mode 1301
18-4. Power Management Register (USBPOWER) in OTG B/Device Mode 1301

Copyright © 2012–2019, Texas Instruments Incorporated

18-5. USB Transmit Interrupt Status Register (USBTXIS) 1303
18-6. USB Receive Interrupt Status Register (USBRXIS) 1305
18-7. USB Transmit Interrupt Status Enable Register (USBTXIE) 1307
18-8. USB Receive Interrupt Enable Register (USBRXIE) 1309
18-9. USB General Interrupt Status Register (USBIS) in OTG A/Host Mode 1311
18-10. USB General Interrupt Status Register (USBIS) in OTG B/Device Mode 1312
18-11. USB Interrupt Enable Register (USBIE) in OTG A/Host Mode 1313
18-12. USB Interrupt Enable Register (USBIE) in OTG B/Device Mode 1314
18-13. Frame Number Register (FRAME) 1315
18-14. USB Endpoint Index Register (USBEPIDX) 1315
18-15. USB Test Mode Register (USBTEST) in OTG A/Host Mode 1316
18-16. USB Test Mode Register (USBTEST) in OTG B/Device Mode 1316
18-17. USB FIFO Endpoint *n* Register (USBFIFO[*n*]) 1318
18-18. USB Device Control Register (USBDEVCTL) 1319
18-19. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ) 1321
18-20. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ) 1322
18-21. USB Transmit FIFO Start Address Register (USBTXFIFOADDR]) 1323
18-22. USB Receive FIFO Start Address Register (USBRXFIFOADDR) 1324
18-23. USB Connect Timing Register (USBCONTIM) 1325
18-24. USB OTG VBUS Pulse Timing Register (USBVPLEN) 1325
18-25. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF) 1326
18-26. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF) 1326
18-27. USB Transmit Functional Address Endpoint *n* Registers (USBTXFUNCADDR[*n*]) 1327
18-28. USB Transmit Hub Address Endpoint *n* Registers (USBTXHUBADDR[*n*]) 1328
18-29. USB Transmit Hub Port Endpoint *n* Registers (USBTXHUBPORT[*n*]) 1329
18-30. USB Receive Functional Address Endpoint *n* Registers (USBFIFO[*n*]) 1330
18-31. USB Receive Hub Address Endpoint *n* Registers (USBRXHUBADDR[*n*]) 1331
18-32. USB Transmit Hub Port Endpoint *n* Registers (USBRXHUBPORT[*n*]) 1332
18-33. USB Maximum Transmit Data Endpoint *n* Registers (USBTXMAXP[*n*]) 1333
18-34. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in OTG A/Host Mode 1334
18-35. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in OTG B/Device Mode 1335
18-36. USB Control and Status Endpoint 0 High Register (USBCSRH0) in OTG A/Host Mode 1336
18-37. USB Control and Status Endpoint 0 High Register (USBCSRH0) in OTG B/Device Mode 1336
18-38. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0) 1337
18-39. USB Type Endpoint 0 Register (USBTYPE0) 1337
18-40. USB NAK Limit Register (USBNAKLMT) 1338
18-41. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in OTG A/Host Mode 1339
18-42. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in OTG B/Device Mode 1340
18-43. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in OTG A/Host Mode 1342
18-44. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in OTG B/Device Mode 1343
18-45. USB Maximum Receive Data Endpoint *n* Registers (USBRXMAXP[*n*]) 1344
18-46. USB Receive Control and Status Endpoint *n* Low Register (USBCSRL[*n*]) in OTG A/Host Mode 1345
18-47. USB Control and Status Endpoint *n* Low Register (USBCSRL[*n*]) in OTG B/Device Mode 1346
18-48. USB Receive Control and Status Endpoint *n* High Register (USBCSRH[*n*]) in OTG A/Host Mode 1348
18-49. USB Control and Status Endpoint *n* High Register (USBCSRH[*n*]) in OTG B/Device Mode 1349
18-50. USB Maximum Receive Data Endpoint *n* Registers (USBRXCOUNT[*n*]) 1350
18-51. USB Host Transmit Configure Type Endpoint n Register (USBTXTYPE[*n*]) 1351
18-52. USB Host Transmit Interval Endpoint n Register (USBTXINTERVAL[*n*]) 1352
18-53. USB Host Configure Receive Type Endpoint n Register (USBRXTYPE[*n*]) 1353

Copyright © 2012–2019, Texas Instruments Incorporated

18-54. USB Host Receive Polling Interval Endpoint n Register (USBRXINTERVAL[*n*]) 1354
18-55. USB Request Packet Count in Block Transfer Endpoint *n* Registers (USBRQPKTCOUNT[*n*]) 1355
18-56. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS) 1356
18-57. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS) 1358
18-58. USB External Power Control Register (USBEPC) 1360
18-59. USB External Power Control Raw Interrupt Status Register (USBEPCRIS) 1362
18-60. USB External Power Control Interrupt Mask Register (USBEPCIM) 1363
18-61. USB External Power Control Interrupt Status and Clear Register (USBEPCISC) 1364
18-62. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) 1365
18-63. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) 1366
18-64. USB Device RESUME Interrupt Status and Clear Register (USBDRISC) 1367
18-65. USB General-Purpose Control and Status Register (USBGPCS) 1368
18-66. USB VBUS Droop Control Register (USBVDC) 1369
18-67. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCRIS) 1370
18-68. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCIM) 1371
18-69. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCISC) 1372
18-70. USB ID Valid Detect Raw Interrupt Status Register (USBIDVRIS) 1373
18-71. USB ID Valid Detect Interrupt Mask Register (USBIDVIM) 1374
18-72. USB ID Valid Detect Interrupt Status and Clear Register (USBIDVISC) 1375
18-73. USB DMA Select Register (USBDMASEL) 1376
19-1. Ethernet MAC 1381
19-2. Ethernet MAC Block Diagram 1382
19-3. Ethernet Frame 1382
19-4. Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK) Register 1388
19-5. Ethernet MAC Interrupt Mask (MACIM) Register 1389
19-6. Ethernet MAC Receive Control (MACRCTL) Register 1390
19-7. Ethernet MAC Transmit Control (MACTCTL) Register 1391
19-8. Ethernet MAC Data (MACDATA) Register (READ) 1393
19-9. Ethernet MAC Data (MACDATA) Register (WRITE) 1393
19-10. Ethernet MAC Individual Address 0 (MACIA0) Register 1394
19-11. Ethernet MAC Individual Address 0 (MACIA1) Register 1395
19-12. Ethernet MAC Threshold (MACTHR) Register 1396
19-13. Ethernet MAC Management Control (MACMCTL) Register 1397
19-14. Ethernet MAC Management Divider (MACMDV) Register 1398
19-15. Ethernet MAC Management Address Register (MACMAR) 1398
19-16. Ethernet MAC Management Transmit Data (MACMTXD) Register 1398
19-17. Ethernet MAC Management Receive Data (MACMRXD) Register 1400
19-18. Ethernet MAC Number of Packets (MACNP) Register 1400
19-19. Ethernet MAC Transmission Request (MACTR) Register 1401
19-20. Ethernet MAC Timer Support (MACTS) Register 1401
19-21. Ethernet PHY Management Register 0 – Control (MR0) Register 1402
19-22. Ethernet PHY Management Register 1 – Status (MR1) Register 1404
19-23. Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2) Register 1405
19-24. Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3) Register 1405
19-25. Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4) Register 1406
19-26. Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5) Register 1407
19-27. Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6) Register 1408
20-1. SSI Block Diagram 1411

Copyright © 2012–2019, Texas Instruments Incorporated

20-2. TI Synchronous Serial Frame Format (Single Transfer) 1413
20-3. TI Synchronous Serial Frame Format (Continuous Transfer) 1414
20-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0 1414
20-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0 1414
20-6. Freescale SPI Frame Format with SPO =0 and SPH=1 1415
20-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0 1416
20-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0 1416
20-9. Freescale SPI Frame Format with SPO =1 and SPH =1 1417
20-10. SSICR0 Register 1422
20-11. SSICR1 Register 1424
20-12. SSIDR Register 1426
20-13. SSISR Register 1427
20-14. SSICPSR Register 1428
20-15. SSIIM Register 1429
20-16. SSIRIS Register 1430
20-17. SSIMIS Register 1432
20-18. SSIICR Register 1434
20-19. SSIDMACTL Register 1435
20-20. SSIPV Register 1436
20-21. SSIPP Register 1437
20-22. SSIPC Register 1438
20-23. SSIPeriphID4 Register 1439
20-24. SSIPeriphID5 Register 1440
20-25. SSIPeriphID6 Register 1441
20-26. SSIPeriphID7 Register 1442
20-27. SSIPeriphID0 Register 1443
20-28. SSIPeriphID1 Register 1444
20-29. SSIPeriphID2 Register 1445
20-30. SSIPeriphID3 Register 1446
20-31. SSIPCellID0 Register 1447
20-32. SSIPCellID1 Register 1448
20-33. SSIPCellID2 Register 1449
20-34. SSIPCellID3 Register 1450
21-1. UART Module Block Diagram 1453
21-2. UART Character Frame 1454
21-3. IrDA Data Modulation 1455
21-4. LIN Message 1456
21-5. LIN Synchronization Field 1457
21-6. UART and SCI Connections for Loopback Mode 1459
21-7. UART Data Register (UARTDR) 1462
21-8. UART Receive Status Register (UARTRSR/UARTECR) 1463
21-9. UART Receive Status/Error Clear Register (UARTRSR/UARTECR) 1464
21-10. UART Flag Register (UARTFR) 1464
21-11. UART IrDA Low-Power Register (UARTILPR) 1466
21-12. UART Integer Baud-Rate Divisor Register (UARTIBRD) 1466
21-13. UART Fractional Baud-Rate Divisor Register (UARTFBRD) 1467
21-14. UART Line Control Register (UARTLCRH) 1467
21-15. UART Control (UARTCTL) Register 1468
21-16. UART Interrupt FIFO Level Select (UARTIFLS) Register 1470

Copyright © 2012–2019, Texas Instruments Incorporated

21-17. UART Interrupt Mask (UARTIM) Register 1471
21-18. UART Raw Interrupt Status (UARTRIS) Register 1473
21-19. UART Masked Interrupt Status (UARTMIS) Register 1475
21-20. UART Interrupt Clear (UARTICR) Register 1477
21-21. UART DMA Control (UARTDMACTL) Register 1478
21-22. UART LIN Control (UARTLCTL) Register 1478
21-23. UART LIN Snap Shot (UARTLSS) Register 1479
21-24. UART LIN Timer (UARTLTIM) Register 1479
21-25. UART Peripheral Identification 4 (UARTPeriphID4) Register 1480
21-26. UART Peripheral Identification 5 (UARTPeriphID5) Register 1480
21-27. UART Peripheral Identification 6 (UARTPeriphID6) Register 1480
21-28. UART Peripheral Identification 7 (UARTPeriphID7) Register 1481
21-29. UART Peripheral Identification 0 (UARTPeriphID0) Register 1481
21-30. UART Peripheral Identification 1 (UARTPeriphID1) Register 1481
21-31. UART Peripheral Identification 2 (UARTPeriphID2) Register 1482
21-32. UART Peripheral Identification 3 (UARTPeriphID3) Register 1482
21-33. UART PrimeCell Identification 0 (UARTPCellID0) Register 1482
21-34. UART PrimeCell Identification 1 (UARTPCellID1) Register 1483
21-35. UART PrimeCell Identification 2 (UARTPCellID2) Register 1483
21-36. UART PrimeCell Identification 3 (UARTPCellID3) Register 1483
22-1. I2C Block Diagram 1485
22-2. I2C Bus Configuration 1486
22-3. START and STOP Conditions 1486
22-4. Complete Data Transfer with a 7-Bit Address 1487
22-5. R/S Bit in First Byte 1487
22-6. Data Validity During Bit Transfer on the I2C Bus 1487
22-7. Master Single TRANSMIT 1491
22-8. Master Single RECEIVE 1492
22-9. Master TRANSMIT with Repeated START 1493
22-10. Master RECEIVE with Repeated START 1494
22-11. Master RECEIVE with Repeated START after TRANSMIT with Repeated START 1495
22-12. Master TRANSMIT with Repeated START after RECEIVE with Repeated START 1495
22-13. Slave Command Sequence 1496
22-14. I2C Master Slave Address (I2CMSA) Register 1499
22-15. I2C Master Control/Status (I2CMCS) (Read-Only) Register 1500
22-16. I2C Master Control/Status (I2CMCS) (Write-Only) Register 1501
22-17. I2C Master Data (I2CMDR) Register 1504
22-18. I2C Master Timer Period (I2CMTPR) Register 1504
22-19. I2C Master Interrupt Mask (I2CMIMR) Register 1505
22-20. I2C Master Raw Interrupt Status (I2CMRIS) Register 1505
22-21. I2C Master Masked Interrupt Status (I2CMMIS) Register 1506
22-22. I2C Master Interrupt Clear (I2CMICR) Register 1506
22-23. I2C Master Configuration (I2CMCR) Register 1507
22-24. I2C Slave Own Address (I2CSOAR) Register 1508
22-25. I2C Slave Control/Status (I2CSCSR) Register (Read-Only) 1508
22-26. I2C Slave Control/Status (I2CSCSR) Register (Write-Only) 1509
22-27. I2C Slave Data (I2CSDR) Register 1509
22-28. I2C Slave Interrupt Mask (I2CSIMR) Register 1509
22-29. I2C Slave Raw Interrupt Status (I2CSRIS) Register 1510

Copyright © 2012–2019, Texas Instruments Incorporated

22-30. I2C Slave Masked Interrupt Status (I2CSMIS) Register 1510
22-31. I2C Slave Interrupt Clear (I2CSICR) Register 1511
23-1. CAN Block Diagram 1514
23-2. CAN_MUXing 1517
23-3. CAN Core in Silent Mode 1517
23-4. CAN Core in Loopback Mode 1518
23-5. CAN Core in External Loopback Mode 1519
23-6. CAN Core in Loopback Combined with Silent Mode 1519
23-7. Initialization of a Transmit Object 1523
23-8. Initialization of a single Receive Object for Data Frames 1524
23-9. Initialization of a single Receive Object for Remote Frames 1524
23-10. CPU Handling of a FIFO Buffer (Interrupt Driven) 1529
23-11. Bit Timing 1530
23-12. The Propagation Time Segment 1531
23-13. Synchronization on Late and Early Edges 1533
23-14. Filtering of Short Dominant Spikes 1534
23-15. Structure of the CAN Core's CAN Protocol Controller 1535
23-16. Data Transfer Between IF1 and IF2 Registers and Message RAM 1539
23-17. Structure of a Message Object 1540
23-18. Message RAM Representation in Debug Mode 1543
23-19. CAN Control Register (CAN CTL) [offset = 0x00] 1544
23-20. Error and Status Register (CAN ES) [offset = 0x04] 1546
23-21. Error Counter Register (CAN ERRC) [offset = 0x08] 1548
23-22. Bit Timing Register (CAN BTR) [offset = 0x0C] 1548
23-23. Interrupt Register (CAN INT) [offset = 0x10] 1549
23-24. Test Register (CAN TEST) [offset = 0x14] 1550
23-25. Parity Error Code Register (CAN PERR) [offset = 0x1C] 1551
23-26. Auto-Bus-On Time Register (CAN ABOTR) [offset = 0x80] 1552
23-27. Transmission Request Register (CAN TXRQ) [offset = 0x88] 1552
23-28. New Data Register (CAN NWDAT) [offset = 0x9C] 1553
23-29. Interrupt Pending Register (CAN INTPND) [offset = 0xB0] 1553
23-30. Message Valid Register (CAN MSGVAL) [offset = 0xC4] 1554
23-31. Interrupt Multiplexer Register (CAN INTMUX) [offset = 0xD8] 1554
23-32. IF1 Command Registers (CAN IF1CMD) [offset = 0x100] 1555
23-33. IF2 Command Registers (CAN IF2CMD) [offset = 0x120] 1555
23-34. IF1 Mask Register (CAN IF1MSK) [offset = 0x104] 1557
23-35. IF2 Mask Register (CAN IF2MSK) [offset = 0x124] 1558
23-36. IF1 Arbitration Register (CAN IF1ARB) [offset = 0x108] 1558
23-37. IF2 Arbitration Register (CAN IF2ARB) [offset = 0x128] 1559
23-38. IF1 Message Control Register (CAN IF1MCTL) [offset = 0x10C] 1560
23-39. IF2 Message Control Register (CAN IF2MCTL) [offset = 0x12C] 1560
23-40. IF1 Data A Register (CAN IF1DATA) [offset = 0x110] 1561
23-41. IF1 Data B Register (CAN IF1DATB) [offset = 0x114] 1561
23-42. IF2 Data A Register (CAN IF2DATA) [offset = 0x130] 1561
23-43. IF2 Data B Register (CAN IF2DATB) [offset = 0x134] 1562
23-44. IF3 Observation Register (CAN IF3OBS) [offset = 0x140] 1562
23-45. IF3 Mask Register (CAN IF3MSK) [offset = 0x144] 1563
23-46. IF3 Arbitration Register (CAN IF3ARB) [offset = 0x148] 1563
23-47. IF3 Message Control Register (CAN IF3MCTL) [offset = 0x14C] 1564

Copyright © 2012–2019, Texas Instruments Incorporated

23-48. IF3 Data A Register (CAN IF3DATA) [offset = 0x150] 1565
23-49. IF3 Data A Register (CAN IF3DATB) [offset = 0x154] 1566
23-50. IF3 Update Enable Register (CAN IF3UPD) [offset = 0x160] 1566
24-1. Cortex-M3 Processor Block Diagram 1569
24-2. Cortex-M3 Register Set 1571
24-3. Cortex General-Purpose Registers 0-12 (R0-R12) 1572
24-4. Stack Pointer Register (SP) 1572
24-5. Link Register 1573
24-6. Program Counter Register 1573
24-7. Program Status Register (PSR) 1574
24-8. Priority Mask Register (PRIMASK) 1576
24-9. Fault Mask Register (FAULTMASK) 1577
24-10. Base Priority Mask Register (BASEPRI) 1577
24-11. Control Register (CONTROL) 1578
24-12. Bit-Band Mapping 1582
24-13. Data Storage 1583
24-14. Vector table 1589
24-15. Exception Stack Frame 1591
25-1. SRD Use Example 1604
25-2. SysTick Control and Status Register (STCTRL) 1609
25-3. SysTick Reload Value Register (STRELOAD) 1610
25-4. SysTick Current Value Register (STCURRENT) 1610
25-5. Interrupt 0-31 Set Enable (EN0) Register 1611
25-6. Interrupt 32-63 Set Enable 1 (EN1) Register 1612
25-7. Interrupt 64-95 Set Enable 2 (EN2) Register 1612
25-8. Interrupt 96-127 Set Enable 3 (EN3) Register 1613
25-9. Interrupt 128-133 Set Enable 4 (EN4) Register 1613
25-10. Interrupt 0-31 Clear Enable (DIS0) Register 1614
25-11. Interrupt 32-63 Clear Enable (DIS1) Register 1614
25-12. Interrupt 64-95 Clear Enable (DIS2) Register 1615
25-13. Interrupt 96-127 Clear Enable (DIS3) Register 1616
25-14. Interrupt 128-133 Clear Enable (DIS4) Register 1617
25-15. Interrupt 0-31 Set Pending (PEND0) Register 1617
25-16. Interrupt 32-63 Set Pending (PEND1) Register 1618
25-17. Interrupt 64-95 Set Pending (PEND2) Register 1618
25-18. Interrupt 96-127 Set Pending (PEND3) Register 1619
25-19. Interrupt 128-133 Set Pending (PEND4) Register 1619
25-20. Interrupt 0-31 Clear Pending (UNPEND0) Register 1620
25-21. Interrupt 32-63 Clear Pending (UNPEND1) Register 1620
25-22. Interrupt 64-95 Clear Pending (UNPEND2) Register 1621
25-23. Interrupt 96-127 Clear Pending (UNPEND3) Register 1622
25-24. Interrupt 128-133 Clear Pending (UNPEND4) Register 1623
25-25. Interrupt 0-31 Active Bit (ACTIVE0) Register 1623
25-26. Interrupt 32-63 Active Bit (ACTIVE1) Register 1624
25-27. Interrupt 64-95 Active Bit (ACTIVE2) Register 1624
25-28. Interrupt 96-127 Active Bit (ACTIVE3) Register 1625
25-29. Interrupt 128-133 Active Bit (ACTIVE4) Register 1625
25-30. Interrupt 0-133 Priority (PRI0-PRI33) Registers 1626
25-31. Software Trigger Interrupt (SWTRIG) Register 1627

Copyright © 2012–2019, Texas Instruments Incorporated

25-32. Auxiliary Control (ACTLR) Register 1628
25-33. CPU ID Base (CPUID) Register 1629
25-34. Interrupt Control and State (INTCTRL) Register 1630
25-35. Vector Table Offset (VTABLE) Register 1633
25-36. Application Interrupt and Reset Control (APINT) Register 1634
25-37. System Control (SYSCTRL) Register 1636
25-38. Configuration and Control (CFGCTRL) Register 1637
25-39. System Handler Priority 1 (SYSPRI1) Register 1638
25-40. System Handler Priority 2 (SYSPRI2) Register 1639
25-41. System Handler Priority 3 (SYSPRI3) Register 1639
25-42. System Handler Control and State (SYSHNDCTRL) Register 1640
25-43. Configurable Fault Status (FAULTSTAT) Register 1643
25-44. Hard Fault Status (HFAULTSTAT) Register 1647
25-45. Memory Management Fault Address (MMADDR) Register 1648
25-46. Bus Fault Address (FAULTADDR) Register Register 1648
25-47. MPU Type (MPUTYPE) Register 1649
25-48. MPU Control (MPUCTRL) Register 1650
25-49. MPU Region Number (MPUNUMBER) Register 1651
25-50. MPU Region Base Address (MPUBASE) Register 1651
25-51. MPU Region Attribute and Size (MPUATTR) Register 1653

Copyright © 2012–2019, Texas Instruments Incorporated

List of Tables

1-1. Signals for System Control and Clocks 80
1-2. Master Subsystem Device Configuration 82
1-3. Device Level Reset Sources 83
1-4. Device Bring-Up Time Line 84
1-5. Master Subsystem Rests, Signals and Effects 89
1-6. Control Subsystem Resets, Signals and Effects 92
1-7. EMU0/1 Pin Values for WIR Mode 93
1-8. Master Subsystem Exceptions 96
1-9. Enabling Interrupt 104
1-10. Interrupt Vector Table Mapping 104
1-11. Vector Table Mapping After Reset Operation 105
1-12. PIE Vector Table Mapping 110
1-13. PIE Vector Table 110
1-14. Access to EALLOW-Protected Registers 119
1-15. Reference Clock Limits for Detecting a Missing Clock 121
1-16. CPU-Timers 0, 1, 2 Configuration and Control Registers 137
1-17. TIMERxTIM Register Field Descriptions 137
1-18. TIMERxTIMH Register Field Descriptions 138
1-19. TIMERxPRD Register Field Descriptions 138
1-20. TIMERxPRDH Register Field Descriptions 138
1-21. TIMERxTCR Register Field Descriptions 138
1-22. TIMERxTPR Register Field Descriptions 139
1-23. TIMERxTPRH Register Field Descriptions 140
1-24. Device Low Power Modes for Active Power Reduction 140
1-25. M3 Subsystem Low-Power Modes 141
1-26. Low-Power Modes Configuration 143
1-27. Master Subsystem Secure RAM Zone Selection 144
1-28. Security Levels 145
1-29. OTPSECLOCK - Reserved Locations in OTP Memory 147
1-30. M3 Zone1 - Reserved Locations in Flash Memory 147
1-31. M3 Zone2 - Reserved Locations in Flash Memory 148
1-32. C28x - Reserved Locations in Flash Memory 148
1-33. Zone Security Status 152
1-34. Zone ECSL Status 155
1-35. IPC MSG RAM Read/Write Accesses 157
1-36. MTOCIPC Message Registers 161
1-37. CTOMIPC Message Registers 161
1-38. System Control, Configuration Registers Address Map 165
1-39. Device Identification 0 (DID0) Register Field Descriptions 173
1-40. Device Identification 1 (DID1) Register Field Descriptions 173
1-41. Device Configuration 1 (DC1) Register Field Descriptions 174
1-42. Device Configuration 2 (DC2) Register Field Descriptions 175
1-43. Device Configuration 4 (DC4) Register Field Descriptions 177
1-44. Device Configuration 6 (DC6) Register Field Descriptions 179
1-45. Device Configuration 10 (DC10) Register Field Descriptions 179
1-46. Device Configuration 7 (DC7) Register Field Descriptions 180
1-47. Master Subsystem Configuration (MCNF) Register Field Descriptions 180

Copyright © 2012–2019, Texas Instruments Incorporated

1-48. Serial Port Loop Back Control (SERPLOOP) Register Field Descriptions 181
1-49. Master Subsystem: ACIB Status (MCIBSTATUS) Register Field Descriptions 181
1-50. C28 Device Part ID (PARTID) Register Field Descriptions 182
1-51. C28 Revision ID (REVID) Register Field Descriptions 182
1-52. Control Subsystem Device Configuration (DEVICECNF) Register Field Descriptions 183
1-53. Control Subsystem Peripheral Configuration 0 (CCNF0) Register Field Descriptions 184
1-54. Control Subsystem Peripheral Configuration 1 (CCNF1) Register Field Descriptions 184
1-55. Control Subsystem Peripheral Configuration 2 (CCNF2) Register Field Descriptions 185
1-56. Control Subsystem Peripheral Configuration 3 (CCNF3) Register Field Descriptions 186
1-57. Control Subsystem Peripheral Configuration 4 (CCNF4) Register Field Descriptions 187
1-58. Master Subsystem Memory Configuration (MEMCNF) Register Field Descriptions 187
1-59. Subsystem Reset Configuration/Control (CRESCNF) Register Field Descriptions 188
1-60. Control Subsystem Reset Status (CRESSTS) Register Field Descriptions 188
1-61. Master Reset Cause (MRESC) Register Field Descriptions 190
1-62. C28 Reset Cause Register (CRESC) Register Field Descriptions 191
1-63. Software Reset Control 0 (SRCR0) Register Field Descriptions 192
1-64. Software Reset Control 1 (SRCR1) Register Field Descriptions 193
1-65. Software Reset Control 2 (SRCR2) Register Field Descriptions 194
1-66. Software Reset Control 3 (SRCR3) Register Field Descriptions 196
1-67. Master Subsystem Wait-In-Reset (MWIR) Register Field Descriptions 197
1-68. C28 Wait-In-Reset (CWIR) Register Field Descriptions 197
1-69. M3NMI Configuration (MNMICFG) Register Field Descriptions 198
1-70. M3NMI Flag (MNMIFLG) Register Field Descriptions 199
1-71. M3NMI Flag Clear (MNMIFLGCLR) Register Field Descriptions 200
1-72. M3NMI Flag Force (MNMIFLGFRC) Register Field Descriptions 201
1-73. M3NMI Watchdog Counter (MNMIWDCNT) Register Field Descriptions 202
1-74. M3NMI Watchdog Period (MNMIWDPRD) Register Field Descriptions 202
1-75. C28 NMI Configuration (CNMICFG) Register Field Descriptions 203
1-76. C28 NMI Flag (CNMIFLG) Register Field Descriptions 204
1-77. C28 NMI Flag Clear (CNMIFLGCLR) Register Field Descriptions 205
1-78. C28 NMI Flag Force (CNMIFLGFRC) Register Field Descriptions 206
1-79. C28 NMI Watchdog Counter (CNMIWDCNT) Register Field Descriptions 206
1-80. C28 NMI Watchdog Period (CNMIWDPRD) Register Field Descriptions 207
1-81. PIE, Control (PIECTRL) Register Field Descriptions 207
1-82. PIE, Acknowledge (PIEACK) Register Field Descriptions 208
1-83. PIE, INTx Group Enable Register (PIEIERx) (x = 1 to 12) Field Descriptions 208
1-84. PIE, INTx Group Flag Register (PIEIFRx) (x = 1 to 12) Field Descriptions 209
1-85. CPU Interrupt Flag Register Field Descriptions (IFR) 210
1-86. CPU Interrupt Enable Register (IER) Field Descriptions 212
1-87. Debug Interrupt Enable Register (DBGIER) Field Descriptions 213
1-88. C28 External Interrupt 1 Configuration Register (XINT1CR) Field Descriptions 215
1-89. C28 External Interrupt 2 Configuration Register (XINT2CR) Field Descriptions 215
1-90. C28 External Interrupt 3 Configuration Register (XINT3CR) Field Descriptions 216
1-91. C28 External Interrupt 1 Counter Register (XINT1CTR) Field Descriptions 216
1-92. C28 External Interrupt 2 Counter Register (XINT2CTR) Field Descriptions 216
1-93. C28 External Interrupt 3 Counter Register (XINT3CTR) Field Descriptions 217
1-94. System PLL Configuration (SYSPLLCTL) Register Field Descriptions 217
1-95. Control Subsystem Clock Disable (CCLKOFF) Register Field Descriptions 218
1-96. M3 Configuration Write Allow (MWRALLOW) Register Field Descriptions 218

Copyright © 2012–2019, Texas Instruments Incorporated

1-97. M3 Configuration Lock (MLOCK) Register Field Descriptions 219
1-98. Missing Clock Status (MCLKSTS) Register Field Descriptions 219
1-99. Missing Clock Force (MCLKFRCCLR) Register Field Descriptions 220
1-100. Missing Clock Enable (MCLKEN) Register Field Descriptions 220
1-101. Missing Clock Reference Limit (MCLKLIMIT) Register Field Descriptions 221
1-102. C28 USER_SWREG1 Register Field Descriptions 221
1-103. C28_USER_SWREG2 Register Field Descriptions 221
1-104. System PLL Multiplier (SYSPLLMULT) Register Field Descriptions 222
1-105. System Clock Divider (SYSDIVSEL) Register Field Descriptions 223
1-106. System PLL Lock Status (SYSPLLSTS) Register Field Descriptions 223
1-107. Master Subsystem Clock Divider (M3SSDIVSEL) Register Field Descriptions 224
1-108. XPLL CLKOUT Control (XPLLCLKCFG) Register Field Descriptions 224
1-109. USB PLL Configuration (UPLLCTL) Register Field Descriptions 225
1-110. USB PLL Multiplier (UPLLMULT) Register Field Descriptions 225
1-111. USB PLL Lock Status (UPLLSTS) Register Field Descriptions 226
1-112. Bit Clock Source Selection for CAN0 (CAN0BCLKSEL) Register Field Descriptions 226
1-113. Bit Clock Source Selection for CAN1 (CAN1BCLKSEL) Register Field Descriptions 227
1-114. Run Mode Clock Configuration (RCC) Register Field Descriptions 227
1-115. Master GPIO High Performance Bus Control (GPIOHBCTL) Register Field Descriptions 228
1-116. Run Mode Clock Gating Control Register 0 (RCGC0) Field Descriptions 228
1-117. Sleep Mode Clock Gating Control Register 0 (SCGC0) Field Descriptions 229
1-118. Deep Sleep Mode Clock Gating Control Register 0 (DCGC0) Field Descriptions 229
1-119. Run Mode Clock Gating Control Register 1 (RCGC1) Field Descriptions 230
1-120. Sleep Mode Clock Gating Control Register 1 (SCGC1) Field Descriptions 231
1-121. Deep Sleep Mode Clock Gating Control Register 1 (DCGC1) Field Descriptions 233
1-122. Run Mode Clock Gating Control Register 2 (RCGC2) Field Descriptions 234
1-123. Sleep Mode Clock Gating Control Register 2 (SCGC2) Field Descriptions 236
1-124. Deep Sleep Mode Clock Gating Control Register 2 (DCGC2) Field Descriptions 238
1-125. Run Mode Clock Gating Control Register 3 (RCGC3) Field Descriptions 239
1-126. Sleep Mode Clock Gating Control Register 3 (SCGC3) Field Descriptions 240
1-127. Deep Sleep Mode Clock Gating Control Register 3 (DCGC3) Field Descriptions 240
1-128. Deep Sleep Clock Configuration (DSLPCLKCFG) Register Field Descriptions 241
1-129. C28 CPU Timer 2 Clock Configuration (CLKCTL) Register Field Descriptions 242
1-130. Peripheral Clock Control Register 0 (PCLKCR0) Register Field Descriptions 243
1-131. Peripheral Clock Control Register 1 (PCLKCR1) Register Field Descriptions 243
1-132. Peripheral Clock Control Register 2 (PCLKCR2) Register Field Descriptions 244
1-133. Peripheral Clock Control Register 3 (PCLKCR3) Register Field Descriptions 245
1-134. High-Speed Clock Prescaler (CHISPCP) Register Field Descriptions 246
1-135. Low-Speed Clock Prescaler (CLOSPCP) Register Field Descriptions 246
1-136. C28 XCLKOUT Divider Register (CXCLK) Field Descriptions 247
1-137. Z1_CSMKEY0 Register Field Descriptions 248
1-138. Z1_CSMKEY1 Register Field Descriptions 248
1-139. Z1_CSMKEY2 Register Field Descriptions 248
1-140. Z1_CSMKEY3 Register Field Descriptions 249
1-141. Z1_ECSLKEY0 Register Field Descriptions 249
1-142. Z1_ECSLKEY1 Register Field Descriptions 249
1-143. Z2_CSMKEY0 Register Field Descriptions 249
1-144. Z2_CSMKEY1 Register Field Descriptions 250
1-145. Z2_CSMKEY2 Register Field Descriptions 250

Copyright © 2012–2019, Texas Instruments Incorporated

1-146. Z2_CSMKEY3 Register Field Descriptions 250
1-147. Z2_ECSLKEY0 Register Field Descriptions 251
1-148. Z2_ECSLKEY1 Register Field Descriptions 251
1-149. Z1_CSMCR Register Field Descriptions 251
1-150. Z2_CSMCR Register Field Descriptions 253
1-151. Z1_GRABSECTR Register Field Descriptions 254
1-152. Z1_GRABRAMR Register Field Descriptions 256
1-153. Z2_GRABSECTR Register Field Descriptions 257
1-154. Z2_GRABRAMR Register Field Descriptions 258
1-155. Z1_EXEONLYR Register Field Descriptions 260
1-156. Z2_EXEONLYR Register Field Descriptions 261
1-157. OTPSECLOCK Register Field Descriptions 262
1-158. CSMKEY0 Register Field Descriptions 264
1-159. CSMKEY1 Register Field Descriptions 264
1-160. CSMKEY2 Register Field Descriptions 264
1-161. CSMKEY3 Register Field Descriptions 265
1-162. CSMCR Register Field Descriptions 265
1-163. ECSLKEY0 Register Field Descriptions 266
1-164. ECSLKEY1 Register Field Descriptions 266
1-165. EXEONLYR Register Field Description 267
1-166. µCRC Register Summary 268
1-167. µCRCCONFIG Register Field Descriptions 269
1-168. µCRCCONTROL Register Field Descriptions 269
1-169. µCRCRES Register Field Descriptions 270
1-170. M3 to C28 IPC Set (MTOCIPCSET) Field Descriptions 270
1-171. M3 to C28 IPC Clear (MTOCIPCCLR) Register Field Descriptions 272
1-172. M3 to C28 Core Flag (MTOCIPCFLG) Register Field Descriptions 274
1-173. M3 to C28 Core IPC Acknowledge (CTOMIPCACK) Register Field Descriptions 276
1-174. C28 to M3 Core IPC Status (CTOMIPCSTS) Register Field Descriptions 278
1-175. M3 Flash Semaphore Field Descriptions 280
1-176. M3 Flash Semaphore Field Descriptions 281
1-177. CTOMIPCSET Register Field Descriptions 282
1-178. CTOMIPCCLR Register Field Descriptions 284
1-179. CTOMIPCFLG Register Field Descriptions 286
1-180. MTOCIPCACK Register Field Descriptions 288
1-181. MTOCIPCSTS Register Field Descriptions 290
1-182. C28 Flash Semaphore Field Descriptions 291
1-183. C28 Clock Semaphore Field Descriptions 292
1-184. MIPCCOUNTERL Register Field Descriptions 293
1-185. MIPCCOUNTERH Register Field Descriptions 293
1-186. CTOMIPCCOM Register Field Descriptions 293
1-187. CTOMIPCADDR Register Field Descriptions 294
1-188. CTOMIPCDATAW Register Field Descriptions 294
1-189. CTOMIPCDATAR Register Field Descriptions 294
1-190. MTOCIPCCOM Register Field Descriptions 295
1-191. MTOCIPCADDR Register Field Descriptions 296
1-192. MTOCIPCDATAW Register Field Descriptions 296
1-193. MTOCIPCDATAR Register Field Descriptions 296
1-194. CTOMIPCBOOTSTS Register Field Descriptions 297

Copyright © 2012–2019, Texas Instruments Incorporated

1-195. MTOCIPCBOOTMODE Register Field Descriptions 297
2-1. Available CCP Pins 300
2-2. General-Purpose Timer Capabilities 300
2-3. 16-Bit Timer With Prescaler Configurations 301
2-4. Timers Register Map 308
2-5. GPTM Configuration (GPTMCFG) Register Field Descriptions 309
2-6. GPTM Timer A Mode (GPTMTAMR) Register Field Descriptions 310
2-7. GPTM Timer B Mode (GPTMTBMR) Register Field Descriptions 311
2-8. GPTM Control (GPTMCTL) Register Field Descriptions 312
2-9. GPTM Interrupt Mask (GPTMIMR) Register Field Descriptions 313
2-10. GPTM Raw Interrupt Status (GPTMRIS) Register Field Descriptions 314
2-11. GPTM Masked Interrupt Status (GPTMMIS) Register Field Descriptions 316
2-12. GPTM Interrupt Clear (GPTMICR) Register Field Descriptions 317
2-13. GPTM Timer A Interval Load (GPTMTAILR) Register Field Descriptions 318
2-14. GPTM Timer A Interval Load (GPTMTBILR) Register Field Descriptions 318
2-15. GPTM Timer A Match (GPTMTAMATCHR) Register Field Descriptions 319
2-16. GPTM Timer B Match (GPTMTBMATCHR) Register Field Descriptions 319
2-17. GPTM Timer A Prescale (GPTMTAPR) Register Field Descriptions 320
2-18. GPTM Timer A Prescale (GPTMTBPR) Register Field Descriptions 320
2-19. GPTM Timer A Prescale Match (GPTMTAPMR) Register Field Descriptions 320
2-20. GPTM Timer B Prescale Match (GPTMTBPMR) Register Field Descriptions 321
2-21. GPTM Timer A (GPTMTAR) Register Field Descriptions 321
2-22. GPTM Timer B (GPTMTBR) Register Field Descriptions 322
2-23. GPTM Timer A Value (GPTMTAV) Register Field Descriptions 322
2-24. GPTM Timer B Value (GPTMTBV) Register Field Descriptions 323
3-1. Watchdog Timers Register Map 327
3-2. Watchdog Load (WDTLOAD) Register Field Descriptions 327
3-3. Watchdog Value (WDTVALUE) Register Field Descriptions 328
3-4. Watchdog Control (WDTCTL) Register Field Descriptions 329
3-5. Watchdog Interrupt Clear (WDTICR) Register Field Descriptions 329
3-6. Watchdog Raw Interrupt Status (WDTRIS) Register Field Descriptions 330
3-7. Watchdog Masked Interrupt Status (WDTMIS) Register Field Descriptions 330
3-8. Watchdog Test (WDTTEST) Register Field Descriptions 331
3-9. Watchdog Lock (WDTLOCK) Register Field Descriptions 331
3-10. Watchdog Peripheral Identification 4 (WDTPeriphID4) Register Field Descriptions 332
3-11. Watchdog Peripheral Identification 5 (WDTPeriphID5) Register Field Descriptions 332
3-12. Watchdog Peripheral Identification 6 (WDTPeriphID6) Register Field Descriptions 332
3-13. Watchdog Peripheral Identification 7 (WDTPeriphID7) Register Field Descriptions 333
3-14. Watchdog Peripheral Identification 0 (WDTPeriphID0) Register Field Descriptions 333
3-15. Watchdog Peripheral Identification 1 (WDTPeriphID1) Register Field Descriptions 333
3-16. Watchdog Peripheral Identification 2 (WDTPeriphID2) Register Field Descriptions 334
3-17. Watchdog Peripheral Identification 3 (WDTPeriphID3) Register Field Descriptions 334
3-18. Watchdog PrimeCell Identification 0 (WDTPCellID0) Register Field Descriptions 334
3-19. Watchdog PrimeCell Identification 1 (WDTPCellID1) Register Field Descriptions 335
3-20. Watchdog PrimeCell Identification 2 (WDTPCellID2) Register Field Descriptions 335
3-21. Watchdog PrimeCell Identification 3 (WDTPCellID3) Register Field Descriptions 335
4-1. GPIO Pins and Alternate Functions 338
4-2. GPIO Pins and Alternate Mode Functions 340
4-3. GPIO Pad Configuration Examples 345

Copyright © 2012–2019, Texas Instruments Incorporated

4-4. GPIO Interrupt Configuration Example 346
4-5. GPIO Register Map 347
4-6. GPIO Data (GPIODATA) Register Field Descriptions 348
4-7. GPIO Direction (GPIODIR) Register Field Descriptions 349
4-8. GPIO Interrupt Sense (GPIOIS) Register Field Descriptions 349
4-9. GPIO Interrupt Both Edges (GPIOIBE) Register Field Descriptions 350
4-10. GPIO Interrupt Event (GPIOIEV) Register Field Descriptions 350
4-11. GPIO Interrupt Mask (GPIOIM) Register Field Descriptions 351
4-12. GPIO Raw Interrupt Status (GPIORIS) Register Field Descriptions 351
4-13. GPIO Masked Interrupt Status (GPIOMIS) Register Field Descriptions 352
4-14. GPIO Interrupt Clear (GPIOICR) Register Field Descriptions 352
4-15. GPIO Alternate Function Select (GPIOAFSEL) Register Field Descriptions 353
4-16. GPIO Open Drain Select (GPIOODR) Register Field Descriptions 354
4-17. GPIO Pull-Up Select (GPIOPUR) Register Field Descriptions 355
4-18. GPIO Digital Enable (GPIODEN) Register Field Descriptions 355
4-19. GPIO Lock (GPIOLOCK) Register Field Descriptions 356
4-20. GPIO Commit (GPIOCR) Register Field Descriptions 357
4-21. GPIO Analog Mode Select (GPIOAMSEL) Register Field Descriptions 357
4-22. GPIO Port Control (GPIOPCTL) Register Field Descriptions 358
4-23. GPIO Alternate Peripheral Select (GPIOAPSEL) Register Field Descriptions 359
4-24. GPIO Core Select (GPIOCSEL) Register Field Descriptions 360
4-25. GPIO Peripheral Identification 4 (GPIOPeriphID4) Register Field Descriptions 361
4-26. GPIO Peripheral Identification 5 (GPIOPeriphID5) Register Field Descriptions 362
4-27. GPIO Peripheral Identification 6 (GPIOPeriphID6) Register Field Descriptions 363
4-28. GPIO Peripheral Identification 7 (GPIOPeriphID7) Register Field Descriptions 364
4-29. GPIO Peripheral Identification 0 (GPIOPeriphID0) Register Field Descriptions 365
4-30. GPIO Peripheral Identification 1 (GPIOPeriphID1) Register Field Descriptions 365
4-31. GPIO Peripheral Identification 2 (GPIOPeriphID2) Register Field Descriptions 366
4-32. GPIO Peripheral Identification 3 (GPIOPeriphID3) Register Field Descriptions 366
4-33. GPIO PrimeCell Identification 0 (GPIOPCellID0) Register Register Field Descriptions 367
4-34. GPIO PrimeCell Identification 1 (GPIOPCellID1) Register Register Field Descriptions 367
4-35. GPIO PrimeCell Identification 2 (GPIOPCellID2) Register Register Field Descriptions 368
4-36. GPIO PrimeCell Identification 3 (GPIOPCellID3) Register Register Field Descriptions 368
4-37. GPIO Control Registers 375
4-38. GPIO Trip Input Select Registers 375
4-39. GPIO Data Registers 378
4-40. Sampling Period 381
4-41. Sampling Frequency 381
4-42. Case 1: Three-Sample Sampling Window Width 382
4-43. Case 2: Six-Sample Sampling Window Width 382
4-44. Default State of Peripheral Input 385
4-45. GPIOA MUX 386
4-46. GPIOB MUX 387
4-47. GPIOC MUX 388
4-48. GPIOE MUX 389
4-49. Analog MUX 389
4-50. GPIO Port A Multiplexing 1 (GPAMUX1) Register Field Descriptions 391
4-51. GPIO Port A MUX 2 (GPAMUX2) Register Field Descriptions 393
4-52. GPIO Port B MUX 1 (GPBMUX1) Register Field Descriptions 395

Copyright © 2012–2019, Texas Instruments Incorporated

4-53. GPIO Port B MUX 2 (GPBMUX2) Register Field Descriptions 397
4-54. GPIO Port C MUX 1 (GPCMUX1) Register Field Descriptions 399
4-55. GPIO Port E MUX 1 (GPEMUX1) Register Field Descriptions 400
4-56. Analog I/O MUX 1 (AIOMUX1) Register Field Descriptions 401
4-57. Analog I/O MUX 2 (AIOMUX2) Register Field Descriptions 402
4-58. GPIO Port A Qualification Control (GPACTRL) Register Field Descriptions 403
4-59. GPIO Port B Qualification Control (GPBCTRL) Register Field Descriptions 404
4-60. GPIO Port C Qualification Control (GPCCTRL) Register Field Descriptions 405
4-61. GPIO Port E Qualification Control (GPECTRL) Register Field Descriptions 405
4-62. GPIO Port A Qualification Select 1 (GPAQSEL1) Register Field Descriptions 406
4-63. GPIO Port A Qualification Select 2 (GPAQSEL2) Register Field Descriptions 406
4-64. GPIO Port B Qualification Select 1 (GPBQSEL1) Register Field Descriptions 407
4-65. GPIO Port B Qualification Select 2 (GPBQSEL2) Register Field Descriptions 407
4-66. GPIO Port C Qualification Select 1 (GPCQSEL1) Register Field Descriptions 408
4-67. GPIO Port E Qualification Select 1 (GPEQSEL1) Register Field Descriptions 409
4-68. GPIO Port A Direction (GPADIR) Register Field Descriptions 410
4-69. GPIO Port B Direction (GPBDIR) Register Field Descriptions 410
4-70. GPIO Port C Direction (GPCDIR) Register Field Descriptions 411
4-71. GPIO Port E Direction (GPEDIR) Register Field Descriptions 411
4-72. GPIO Port E Pullup Disable (GPEPUD) Register Field Descriptions 412
4-73. Analog I/O DIR (AIODIR) Register Field Descriptions 412
4-74. GPIO Port A Data (GPADAT) Register Field Descriptions 413
4-75. GPIO Port B Data (GPBDAT) Register Field Descriptions 414
4-76. GPIO Port C Data (GPCDAT) Register Field Descriptions 415
4-77. GPIO Port E Data (GPEDAT) Register Field Descriptions 416
4-78. Analog I/O DAT (AIODAT) Register Field Descriptions 417
4-79. GPIO Port A Set (GPASET) Register Field Descriptions 418
4-80. GPIO Port A Clear (GPACLEAR) Register Field Descriptions 418
4-81. GPIO Port A Toggle (GPATOGGLE) Register Field Descriptions 418
4-82. GPIO Port B Set (GPBSET) Register Field Descriptions 419
4-83. GPIO Port B Clear (GPBCLEAR) Register Field Descriptions 419
4-84. GPIO Port B Toggle (GPBTOGGLE) Register Field Descriptions 419
4-85. GPIO Port C Set (GPCSET) Register Field Descriptions 420
4-86. GPIO Port C Clear (GPCCLEAR) Register Field Descriptions 420
4-87. GPIO Port C Toggle (GPCTOGGLE) Register Field Descriptions 420
4-88. GPIO Port E Set (GPESET) Register Field Descriptions 421
4-89. GPIO Port E Clear (GPECLEAR) Register Field Descriptions 421
4-90. GPIO Port E Toggle (GPETOGGLE) Register Field Descriptions 421
4-91. Analog I/O Set (AIOSET) Register Field Descriptions 422
4-92. Analog I/O Clear (AIOCLEAR) Register Field Descriptions 422
4-93. Analog I/O Toggle (AIOTOGGLE) Register Field Descriptions 422
4-94. GPIO Trip Input Select Register (GPTRIPxSEL) Field Descriptions 423
4-95. GPTRIP Input Signals 423
4-96. GPIO Low Power Mode Wakeup Select 1 (GPIOLPMSEL1) Register Field Descriptions 424
4-97. GPIO Low Power Mode Wakeup Select 2 (GPIOLPMSEL2) Register Field Descriptions 424
5-1. Master access for Sx RAM (assuming all other protections are disabled) 427
5-2. Error Handling in Different Scenarios 432
5-3. Mapping of ECC bits in Read Data from ECC/Parity Address Map 433
5-4. Mapping of Parity bits in Read Data from ECC/Parity Address Map 433

Copyright © 2012–2019, Texas Instruments Incorporated

5-5. M3 RAM Configuration Registers Summary 434
5-6. M3 RAM Error Registers Summary 434
5-7. C28x RAM Configuration Registers Summary 435
5-8. C28x RAM Error Registers Summary 435
5-9. Cx DEDRAM Configuration Register 1 (CxDRCR1) Field Descriptions 437
5-10. Cx SHRAM Configuration Register 1 (CxSRCR1) Field Descriptions 438
5-11. Sx SHRAM Master Select Register (MSxMSEL) Field Descriptions 439
5-12. M3 Sx SHRAM Configuration Register 1 (MSxSRCR1) Field Descriptions 440
5-13. M3 Sx SHRAM Configuration Register 2 (MSxSRCR2) Field Descriptions 442
5-14. M3TOC28_MSG_RAM Configuration Register (MTOCMSGRCR) Field Descriptions 444
5-15. Cx RAM Test and Initialization Register 1 (CxRTESTINIT1) Field Descriptions 445
5-16. M3 Sx RAM Test and Initialization Register 1 (MSxRTESTINIT1) Field Descriptions 446
5-17. MTOC_MSG_RAM Test and Initialization Register (MTOCRTESTINIT) Field Descriptions 448
5-18. Cx RAM INITDONE Register 1 (CxRINITDONE1) Field Descriptions 449
5-19. M3 Sx RAM INITDONE Register 1 (MSxRINITDONE1) Field Descriptions 450
5-20. MTOC_MSG_RAM INITDONE Register (MTOCRINITDONE) Field Descriptions 452
5-21. M3 CPU Uncorrectable Write Error Address Register (MCUNCWEADDR) Field Descriptions 452
5-22. M3 µDMA Uncorrectable Write Error Address Register (MDUNCWEADDR) Field Descriptions 452
5-23. M3 CPU Uncorrectable Read Error Address Register (MCUNCREADDR) Field Descriptions 453
5-24. M3 µDMA Uncorrectable Read Error Address Register (MDUNCREADDR) Field Descriptions 453
5-25. M3 CPU Corrected Read Error Address Register (MCPUCREADDR) Field Descriptions 454
5-26. M3 µDMA Corrected Read Error Address Register (MDMACREADDR) Field Descriptions 454
5-27. M3 Uncorrectable Error Flag Register (MUEFLG) Field Descriptions 455
5-28. M3 Uncorrectable Error Force Register (MUEFRC) Field Descriptions 456
5-29. M3 Uncorrectable Error Flag Clear Register (MUECLR) Field Descriptions 457
5-30. M3 Corrected Error Counter Register (MCECNTR) Field Descriptions 457
5-31. M3 Corrected Error Threshold Register (MCETRES) Field Descriptions 458
5-32. M3 Corrected Error Threshold Exceeded Flag Register (MCEFLG) Field Descriptions 458
5-33. M3 Corrected Error Threshold Exceeded Force Register (MCEFRC) Field Descriptions 458
5-34. M3 Corrected Error Threshold Exceeded Flag Clear Register (MCECLR) Field Descriptions 459
5-35. M3 Single Error Interrupt Enable Register (MCEIE) Field Descriptions 459
5-36. Non-Master Access Violation Flag Register (MNMAVFLG) Field Descriptions 460
5-37. Non-Master Access Violation Flag Clear Register (MNMAVCLR) Field Descriptions 460
5-38. Master Access Violation Flag Register (MMAVFLG) Field Descriptions 461
5-39. Master Access Violation Flag Clear Register (MMAVCLR) Field Descriptions 462
5-40. Non-Master CPU Write Access Violation Address Register (MNMWRAVADDR) Field Descriptions 462
5-41. Non-Master DMA Write Access Violation Address Register (MNMDMAWRAVADDR) Field Descriptions 463
5-42. Non-Master CPU Fetch Access Violation Address Register (MNMFAVADDR) Field Descriptions 463
5-43. Master CPU Write Access Violation Address Register (MMWRAVADDR) Field Descriptions 463
5-44. Master DMA Write Access Violation Address Register (CMDMAWRAVADDR) Field Descriptions 464
5-45. Master CPU Fetch Access Violation Address Register (MMFAVADDR) Field Descriptions 464
5-46. Lx DEDRAM Configuration Register 1 (LxDRCR1) Field Descriptions 465
5-47. Lx SHRAM Configuration Register 1 (LxSRCR1) Field Descriptions 466
5-48. C28x Sx SHRAM Master Select Register (CSxMSEL) Field Descriptions 467
5-49. C28x Sx SHRAM Configuration Register 1 (CSxSRCR1) Field Descriptions 468
5-50. C28x Sx SHRAM Configuration Register 2 (CSxSRCR2) Field Descriptions 470
5-51. C28TOC28_MSG_RAM Configuration Register (CTOMMSGRCR) Field Descriptions 471
5-52. M0, M1 and C28T0C28_MSG_RAM Test and Initialization Register (C28RTESTINIT) Field Descriptions 472
5-53. Lx RAM Test and Initialization Register 1 (CLxRTESTINIT1) Field Descriptions 473

Copyright © 2012–2019, Texas Instruments Incorporated

5-54. C28x Sx RAM Test and Initialization Register 1 (CSxRTESTINIT1) Field Descriptions 474
5-55. M0, M1 and C28T0M3_MSG_RAM INIT Done Register (C28RINITDONE) Field Descriptions 476
5-56. C28x Lx RAM_INIT_DONE Register 1 (CLxRINITDONE1) Field Descriptions 477
5-57. C28x Sx RAM_INIT_DONE Register 1 (CSxRINITDONE1) Field Descriptions 478
5-58. C28x CPU Uncorrectable Read Error Address Register (CCUNCREADDR) Field Descriptions 480
5-59. C28x DMA Uncorrectable Read Error Address Register (CDUNCREADDR) Field Descriptions 480
5-60. C28x CPU Corrected Read Error Address Register (CCPUCREADDR) Field Descriptions 480
5-61. C28x DMA Corrected Read Error Address Register (CDMACREADDR) Field Descriptions 481
5-62. C28x Uncorrectable Error Flag Register (CUEFLG) Field Descriptions 481
5-63. C28x Uncorrectable Error Force Register (CUEFRC) Field Descriptions 482
5-64. C28x Uncorrectable Error Flag Clear Register (CUECLR) Field Descriptions 482
5-65. C28x Corrected Error Counter Register (CCECNTR) Field Descriptions 483
5-66. C28x Corrected Error Threshold Register (CCETRES) Field Descriptions 483
5-67. C28x Corrected Error Threshold Exceeded Flag Register (CCEFLG) Field Descriptions 484
5-68. C28x Corrected Error Threshold Exceeded Force Register (CCEFRC) Field Descriptions 484
5-69. C28x Corrected Error Threshold Exceeded Flag Clear Register (CCECLR) Field Descriptions 485
5-70. C28x Single Error Interrupt Enable Register (CCEIE) Field Descriptions 485
5-71. Non-Master Access Violation Flag Register (CNMAVFLG) Field Descriptions 486
5-72. Non-Master Access Violation Force Register (CNMAVFRC) Field Descriptions 487
5-73. Non-Master Access Violation Flag Clear Register (CNMAVCLR) Field Descriptions 487
5-74. Master Access Violation Flag Register (CMAVFLG) Field Descriptions 488
5-75. Master Access Violation Force Register (CMAVFRC) Field Descriptions 488
5-76. Master Access Violation Flag Clear Register (CMAVCLR) Field Descriptions 490
5-77. Non-Master CPU Write Access Violation Address Register (CNMWRAVADDR) Field Descriptions 491
5-78. Non-Master DMA Write Access Violation Address Register (CNMDMAWRAVADDR) Field Descriptions 491
5-79. Non-Master CPU Fetch Access Violation Address Register (CNMFAVADDR) Field Descriptions 491
5-80. Master CPU Write Access Violation Address Register (CMWRAVADDR) Field Descriptions 492
5-81. Master DMA Write Access Violation Address Register (CMDMAWRAVADDR) Field Descriptions 492
5-82. Master CPU Fetch Access Violation Address Register (CMFAVADDR) Field Descriptions 492
5-83. Programmable OTP Locations in M3 OTP 494
5-84. Flash Registers Memory Map on Master Subsystem 508
5-85. Flash Registers Memory Map on Control Subsystem 509
5-86. Flash Read Control Register (FRDCNTL) Field Descriptions 512
5-87. Flash Read Margin Control Register (FSPRD) Field Descriptions 512
5-88. Flash Bank Access Control Register (FBAC) Field Descriptions 513
5-89. Flash Bank Fallback Power Register (FBFALLBACK) Field Description 513
5-90. Flash Bank Pump Control Register (FBPRDY) Field Descriptions 514
5-91. Flash Bank Pump Control Register 1 (FPAC1) Field Descriptions 514
5-92. Flash Bank Pump Control Register 2 (FPAC2) Field Descriptions 515
5-93. Flash Module Access Control Register (FMAC) Field Descriptions 515
5-94. SECZONEREQUEST(SEM) Register Field Descriptions 516
5-95. Flash Read Interface Control Register (FRD_INTF_CTRL) Field Descriptions 517
5-96. ECC Enable Register (ECC_Enable) Field Descriptions 518
5-97. Single Error Address Register (SINGLE_ERR_ADDR) Field Descriptions 518
5-98. Uncorrectable Error Address Register (UNC_ERR_ADDR) Field Descriptions 518
5-99. Error Status Register (ERR_STATUS) Field Descriptions 519
5-100. Error Position Register (ERR_POS) Field Descriptions 519
5-101. Error Status Clear Register (ERR_STATUS_CLR) Field Descriptions 520
5-102. Error Counter Register (ERR_CNT) Field Descriptions 520

Copyright © 2012–2019, Texas Instruments Incorporated

5-103. Error Threshold Register (ERR_THRESHOLD) Field Descriptions 521
5-104. Error Interrupt Flag Register (ERR_INTFLG) Field Descriptions 521
5-105. Error Interrupt Flag Clear Register (ERR_INTCLR) Field Descriptions 522
5-106. Data High Test Register (FDATAH_TEST) Field Descriptions 522
5-107. Data Low Test Register (FDATAL_TEST) Field Descriptions 522
5-108. ECC Test Address Register (FADDR_TEST) Field Descriptions 523
5-109. ECC Test Register (FECC_TEST) Field Descriptions 523
5-110. ECC Control Register (FECC_CTRL) Field Descriptions 523
5-111. Test Data Out High Register (FECC_FOUTH_TEST) Field Descriptions 524
5-112. Test Data Out Low Register (FECC_FOUTL_TEST) Field Descriptions 524
5-113. ECC Status Register (FECC_STATUS) Field Descriptions 524
5-114. Flash Read Control Register (FRDCNTL) Field Descriptions 525
5-115. Flash Read Margin Control Register (FSPRD) Field Descriptions 525
5-116. Flash Bank Access Control Register (FBAC) Field Descriptions 526
5-117. Flash Bank Fallback Power Register (FBFALLBACK) Field Descriptions 526
5-118. Flash Bank Pump Control Register (FBPRDY) Field Descriptions 527
5-119. Flash Bank Pump Control Register 1 (FPAC1) Field Descriptions 527
5-120. Flash Bank Pump Control Register 2 (FPAC2) Field Descriptions 528
5-121. Flash Module Access Control Register (FMAC) Field Descriptions 528
5-122. Flash Read Interface Control Register (FRD_INTF_CTRL) Field Descriptions 529
5-123. ECC Enable Register (ECC_ENABLE) Field Descriptions 529
5-124. SIngle Error Address Register (SINGLE_ERR_ADDR) Field Descriptions 529
5-125. Uncorrectable Error Address Register (UNC_ERR_ADDR) Field Descriptions 529
5-126. Error Status Register (ERR_STATUS) Field Descriptions 530
5-127. Error Position Register (ERR_POS) Field Descriptions 530
5-128. Error Status Clear Register (ERR_STATUS_CLR) Field Descriptions 531
5-129. Error Counter Register (ERR_CNT) Field Descriptions 531
5-130. Error Threshold Register (ERR_THRESHOLD) Field Descriptions 532
5-131. Error Interrupt Flag Register (ERR_INTFLG) Field Descriptions 532
5-132. Error Interrupt Flag Clear Register (ERR_INTCLR) Field Descriptions 532
5-133. Data High Test Register (FDATAH_TEST) Field Descriptions 533
5-134. Data Low Test Register (FDATAL_TEST) Field Descriptions 533
5-135. ECC Test Address Register (FADDR_TEST) Field Descriptions 533
5-136. ECC Test Register (FECC_TEST) Field Descriptions 534
5-137. ECC Control Register (FECC_CTRL) Field Descriptions 534
5-138. Test Data Out High Register (FECC_FOUTH_TEST) Field Descriptions 534
5-139. Test Data Out Low Register (FECC_FOUTL_TEST) Field Descriptions 535
5-140. ECC Status Register (FECC_STATUS) Field Descriptions 535
6-1. Master Subsystem Boot Mode Selection 538
6-2. M-Boot ROM Vector Table 541
6-3. M-Boot ROM Version and Checksum Information 542
6-4. REV0 – User Configurable DCSM OTP Fields 543
6-5. REVA – User Configurable DCSM OTP Fields 543
6-6. M-Boot ROM Entry Points 544
6-7. M-Boot ROM Clock Settings 546
6-8. M-Boot ROM Boot Mode GPIO Assignments 546
6-9. M-Boot ROM Reset Cause Handling 553
6-10. M-Boot ROM Exceptions Handling 553
6-11. M-Boot ROM Serial Boot Commands 556

Copyright © 2012–2019, Texas Instruments Incorporated

6-12. M-Boot ROM CAN Boot Commands 562
6-13. Parallel GPIO Boot 8-Bit Data Stream 565
6-14. C-Boot ROM Version and Checksum Information 574
6-15. C-Boot ROM PIE Mismatch Handler 574
6-16. C-Boot ROM CPU Vector Table 575
6-17. PIE Vector Table in C-Boot ROM 576
6-18. C-Boot ROM Boot Modes 578
6-19. C-Boot ROM Entry Point 579
6-20. C-Boot ROM GPIO Assignments for Boot Modes 581
6-21. MTOC IPC Commands 588
6-22. C-Boot ROM NAK/ERROR Status Returns for MTOCIPCCOM 590
6-23. C-Boot ROM Boot Status Values 591
6-24. CTOM IPC Messages 592
6-25. C-Boot ROM Exceptions Handling 593
6-26. General Structure Of Source Program Data Stream In 16-Bit Mode 596
6-27. LSB/MSB Loading Sequence in 8-Bit Data Stream 598
6-28. SPI 8-Bit Data Stream 604
6-29. I2C 8-Bit Data Stream 609
6-30. Parallel GPIO Boot 8-Bit Data Stream 611
6-31. Bootloader Options 626
7-1. ePWM Module Control and Status Register Set Grouped by Submodule 637
7-2. ePWM Module Control and Status Register Set Grouped by Submodule (Upper Page) 639
7-3. Submodule Configuration Parameters 641
7-4. Time-Base Submodule Registers 644
7-5. Key Time-Base Signals 645
7-6. Counter-Compare Submodule Registers 654
7-7. Counter-Compare Submodule Key Signals 655
7-8. Action-Qualifier Submodule Registers 661
7-9. Action-Qualifier Submodule Possible Input Events 662
7-10. Action-Qualifier Event Priority for Up-Down-Count Mode 664
7-11. Action-Qualifier Event Priority for Up-Count Mode 664
7-12. Action-Qualifier Event Priority for Down-Count Mode 664
7-13. Behavior if CMPA/CMPB is Greater than the Period 664
7-14. Dead-Band Generator Submodule Registers 678
7-15. Classical Dead-Band Operating Modes 679
7-16. Additional Dead-Band Operating Modes 680
7-17. Dead-Band Delay Values in μS as a Function of DBFED and DBRED 682
7-18. PWM-Chopper Submodule Registers 683
7-19. Possible Pulse Width Values for SYSCLKOUT = 80 MHz 685
7-20. Trip-Zone Submodule Registers 688
7-21. Possible Actions On a Trip Event 690
7-22. TRIGxSEL Trigger Options 693
7-23. Event-Trigger Submodule Registers 695
7-24. Digital Compare Submodule Registers 701
7-25. Time-Base Period and Mirror 2 Register (TBPRD / TBPRDM2) Field Descriptions 733
7-26. Time-Base Period High-Resolution and Mirror 2 Register (TBPRDHR / TBPRDHRM2) Field Descriptions . 733
7-27. Time-Base Period Mirror Register (TBPRDM) Field Descriptions 734
7-28. Time-Base Period High-Resolution Mirror Register (TBPRDHRM) Field Descriptions 734
7-29. Time-Base Phase Register and Mirror Register (TBPHS / TBPHSM) Field Descriptions 735

Copyright © 2012–2019, Texas Instruments Incorporated

7-30. Time-Base Phase High-Resolution Register and Mirror Register (TBPHSHR / TBPHSHRM) Field Descriptions 735
7-31. Time-Base Counter Register (TBCTR) Field Descriptions 735
7-32. Time-Base Control Register (TBCTL) Field Descriptions 736
7-33. Time-Base Status Register (TBSTS) Field Descriptions 738
7-34. High-Resolution Period Control Register (HRPCTL) Field Descriptions 738
7-35. Time-Base Control Register 2 (TBCTL2) Field Descriptions 739
7-36. EPWMx Link Register (EPWMXLINK) Field Descriptions 740
7-37. Counter-Compare Control Register (CMPCTL) Field Descriptions 742
7-38. Counter-Compare Control Register (CMPCTL2) Field Descriptions 743
7-39. Compare A High-Resolution and Mirror 2 Register (CMPAHR / CMPAHRM2) Field Descriptions 744
7-40. Counter-Compare A and Mirror 2 Register (CMPA / CMPAM2) Field Descriptions 745
7-41. Compare A High-Resolution Mirror Register (CMPAHRM) Field Descriptions 745
7-42. Counter-Compare A Mirror Register (CMPAM) Field Descriptions 746
7-43. Counter-Compare B Register (CMPBM) Field Descriptions 746
7-44. Counter-Compare B Register (CMPB) Field Descriptions 747
7-45. Counter-Compare C Register (CMPC) Field Descriptions 747
7-46. Counter-Compare D Register (CMPD) Field Descriptions 748
7-47. Compare B High-Resolution Register (CMPBHR) Field Descriptions 748
7-48. Compare B High-Resolution Mirror Register (CMPBHRM) Field Descriptions 748
7-49. Action-Qualifier Output A Control Register and Mirror Register (AQCTLA / AQCTLAM) Field Descriptions 749
7-50. Action-Qualifier Output B Control Register and Mirror Register (AQCTLB / AQCTLBM) Field Descriptions 750
7-51. Action-Qualifier Software Force Register and Mirror Register (AQSFRC / AQSFRCM) Field Descriptions .. 751
7-52. Action-Qualifier Continuous Software Force Register and Mirror Register (AQCSFRC / AQCSFRCM) Field Descriptions 752
7-53. Action Qualifier Control Register (AQCTLR) Field Description 753
7-54. Dead-Band Generator Control Register (DBCTL) Field Descriptions 754
7-55. Dead-Band Generator Rising Edge Delay and Mirror Register (DBRED / DBREDM) Field Descriptions 756
7-56. Dead-Band Generator Falling Edge Delay and Mirror Register (DBFED / DBFEDM) Field Descriptions 756
7-57. Dead Band Rising Edge Delay High-Resolution Register (DBREDHR) Field Descriptions 756
7-58. Dead Band Falling Edge Delay High-Resolution Register (DBFEDHR) Field Descriptions 757
7-59. PWM-Chopper Control Register (PCCTL) Bit Descriptions 758
7-60. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions 760
7-61. Trip-Zone Control Register Field Descriptions 761
7-62. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions 762
7-63. Trip-Zone Flag Register (TZFLG) Field Descriptions 763
7-64. Trip-Zone Clear Register and Mirror Register (TZCLR / TZCLRM) Field Descriptions 764
7-65. Trip-Zone Force Register (TZFRC) Field Descriptions 765
7-66. Trip Zone Digital Compare Event Select Register (TZDCSEL) Field Descriptions 766
7-67. Digital Compare Trip Select (DCTRIPSEL) Field Descriptions 767
7-68. Digital Compare A Control Register (DCACTL) Field Descriptions 768
7-69. Digital Compare B Control Register (DCBCTL) Field Descriptions 769
7-70. Digital Compare Filter Control Register (DCFCTL) Field Descriptions 769
7-71. Digital Compare Capture Control Register (DCCAPCTL) Field Descriptions 770
7-72. Digital Compare Counter Capture Register (DCCAP) Field Descriptions 771
7-73. Digital Compare Filter Offset Register (DCFOFFSET) Field Descriptions 771
7-74. Digital Compare Filter Offset Counter Register (DCFOFFSETCNT) Field Descriptions 771
7-75. Digital Compare Filter Window Register (DCFWINDOW) Field Descriptions 772
7-76. Digital Compare Filter Window Counter Register (DCFWINDOWCNT) Field Descriptions 772

Copyright © 2012–2019, Texas Instruments Incorporated

7-77. Digital Compare A High Trip Input Select (DCAHTRIPSEL) Field Descriptions 773
7-78. Digital Compare A Low Trip Input Select (DCALTRIPSEL) Field Descriptions 774
7-79. Digital Compare B High Trip Input Select (DCBHTRIPSEL) Field Descriptions 775
7-80. Digital Compare B Low Trip Input Select (DCBLTRIPSEL) Field Descriptions 776
7-81. GPTRIP Input Signals 778
7-82. GPIOTRIP Input Select Registers 778
7-83. GPIO Trip Input Select Register (GPTRIPxSEL) Field Descriptions 779
7-84. Event-Trigger Selection Register (ETSEL) Field Descriptions 780
7-85. Event-Trigger Prescale Register (ETPS) Field Descriptions 782
7-86. Event-Trigger Interrupt Pre-Scale Register (ETINTPS) Field Descriptions 783
7-87. Event-Trigger SOC Pre-Scale Register (ETSOCPS) Field Descriptions 784
7-88. Event-Trigger Flag Register (ETFLG) Field Descriptions 785
7-89. Event-Trigger Clear Register and Mirror Register (ETCLR / ETCLRM) Field Descriptions 786
7-90. Event-Trigger Force Register (ETFRC) Field Descriptions 786
7-91. Event-Trigger Counter Initialization Control Register (ETCNTINITCTL) Field Descriptions 787
7-92. Event-Trigger Counter Initialization Register (ETCNTINIT) Field Descriptions 788
8-1. Time-Stamp Counter Register (TSCTR) Field Descriptions 801
8-2. Counter Phase Control Register (CTRPHS) Field Descriptions 801
8-3. Capture-1 Register (CAP1) Field Descriptions 801
8-4. Capture-2 Register (CAP2) Field Descriptions 801
8-5. Capture-3 Register (CAP3) Field Descriptions 802
8-6. Capture-4 Register (CAP4) Field Descriptions 802
8-7. ECAP Control Register 1 (ECCTL1) Field Descriptions 802
8-8. ECAP Control Register 2 (ECCTL2) Field Descriptions 804
8-9. ECAP Interrupt Enable Register (ECEINT) Field Descriptions 806
8-10. ECAP Interrupt Flag Register (ECFLG) Field Descriptions 807
8-11. ECAP Interrupt Clear Register (ECCLR) Field Descriptions 808
8-12. ECAP Interrupt Forcing Register (ECFRC) Field Descriptions 808
8-13. Control and Status Register Set 809
9-1. EQEP Memory Map 824
9-2. Quadrature Decoder Truth Table 827
9-3. eQEP Decoder Control (QDECCTL) Register Field Descriptions 840
9-4. eQEP Control (QEPCTL) Register Field Descriptions 842
9-5. eQEP Position-compare Control (QPOSCTL) Register Field Descriptions 843
9-6. eQEP Capture Control (QCAPCTL) Register Field Descriptions 844
9-7. eQEP Position Counter (QPOSCNT) Register Field Descriptions 844
9-8. eQEP Position Counter Initialization (QPOSINIT) Register Field Descriptions 845
9-9. eQEP Maximum Position Count (QPOSMAX) Register Field Descriptions 845
9-10. eQEP Position-compare (QPOSCMP) Register Field Descriptions 845
9-11. eQEP Index Position Latch (QPOSILAT) Register Field Descriptions 845
9-12. eQEP Strobe Position Latch (QPOSSLAT) Register Field Descriptions 846
9-13. eQEP Position Counter Latch (QPOSLAT) Register Field Descriptions 846
9-14. eQEP Unit Timer (QUTMR) Register Field Descriptions 846
9-15. eQEP Unit Period (QUPRD) Register Field Descriptions 846
9-16. eQEP Watchdog Timer (QWDTMR) Register Field Descriptions 847
9-17. eQEP Watchdog Period (QWDPRD) Register Field Description 847
9-18. eQEP Interrupt Enable(QEINT) Register Field Descriptions 847
9-19. eQEP Interrupt Flag (QFLG) Register Field Descriptions 848
9-20. eQEP Interrupt Clear (QCLR) Register Field Descriptions 849

Copyright © 2012–2019, Texas Instruments Incorporated

9-21. eQEP Interrupt Force (QFRC) Register Field Descriptions 850
9-22. eQEP Status (QEPSTS) Register Field Descriptions 851
9-23. eQEP Capture Timer (QCTMR) Register Field Descriptions 852
9-24. eQEP Capture Period Register (QCPRD) Register Field Descriptions 852
9-25. eQEP Capture Timer Latch (QCTMRLAT) Register Field Descriptions 853
9-26. eQEP Capture Period Latch (QCPRDLAT) Register Field Descriptions 853
10-1. Simplified ACIB Signals 856
10-2. TRIGxSEL Trigger Options 862
10-3. Sample timings with different values of ACQPS 863
10-4. ADC Registers 872
10-5. ADC Configuration and Control Registers (AdcRegs and AdcResult): 872
10-6. ADC Control Register 1 (ADCCTL1) Field Descriptions 873
10-7. ADC Control Register 2 (ADCCTL2) Field Descriptions 875
10-8. ADC Interrupt Flag Register (ADCINTFLG) Field Descriptions 875
10-9. ADC Interrupt Flag Clear Register (ADCINTFLGCLR) Field Descriptions 876
10-10. ADC Interrupt Overflow Register (ADCINTOVF) Field Descriptions 876
10-11. ADC Interrupt Overflow Clear Register (ADCINTOVFCLR) Field Descriptions 877
10-12. INTSELxNy Register Field Descriptions 878
10-13. SOCPRICTL Register Field Descriptions 879
10-14. ADC Sample Mode Register (ADCSAMPLEMODE) Field Descriptions 881
10-15. ADC Interrupt Trigger SOC Select 1 Register (ADCINTSOCSEL1) Register Field Descriptions 882
10-16. ADC Interrupt Trigger SOC Select 2 Register (ADCINTSOCSEL2) Field Descriptions 883
10-17. ADC SOC Flag 1 Register (ADCSOCFLG1) Field Descriptions 883
10-18. ADC SOC Force 1 Register (ADCSOCFRC1) Field Descriptions 884
10-19. ADC SOC Overflow 1 Register (ADCSOCOVF1) Field Descriptions 884
10-20. ADC SOC Overflow Clear 1 Register (ADCSOCOVFCLR1) Field Descriptions 884
10-21. ADC SOC0 - SOC15 Control Registers (ADCSOCxCTL) Register Field Descriptions 885
10-22. ADC Reference/Gain Trim Register (ADCREFTRIM) Field Descriptions 887
10-23. ADC Offset Trim Register (ADCOFFTRIM) Field Descriptions 888
10-24. ADC Revision Register (ADCREV) Field Descriptions 888
10-25. ADC RESULT0 - ADCRESULT15 Registers (ADCRESULTx) Field Descriptions 888
10-26. Analog Subsystem Control Registers (AnalogSysctrlReg) 889
10-27. ADC Interrupt Overflow Detect Register (INTOVF) Field Descriptions 890
10-28. ADC Interrupt Overflow Clear Register (INTOVFCLR) Field Descriptions 891
10-29. Control System: Lock Register (CLOCK) Field Descriptions 892
10-30. Control System: ACIB Status Register (CCIBSTATUS) Field Descriptions 893
10-31. Control System: Clock Control Register (CCLKCTL) Field Descriptions 894
10-32. ADC Start of Conversion Trigger Overflow Detect Register (TRIGOVF) Field Descriptions 895
10-33. ADC Start of Conversion Trigger Overflow Flag Clear Register (TRIGOVFCLR) Field Descriptions 896
10-34. ADC Start of Conversion Trigx Input Select Register (TRIGxSEL) Field Descriptions 897
10-35. Comparator Truth Table 902
10-36. Comparator Module Registers 904
10-37. COMPCTL Register Field Descriptions 904
10-38. Compare Output Status (COMPSTS) Register Field Descriptions 905
10-39. DAC Value (DACVAL) Register Field Descriptions 905
10-40. DAC Test (DACTEST) Register Field Descriptions 906
11-1. Peripheral Interrupt Trigger Source Options 915
11-2. DMA Register Summary 926
11-3. DMA Control Register (DMACTRL) Field Descriptions 927

Copyright © 2012–2019, Texas Instruments Incorporated

11-4. Debug Control Register (DEBUGCTRL) Field Descriptions 929
11-5. Revision Register (REVISION) Field Descriptions 929
11-6. Priority Control Register 1 (PRIORITYCTRL1) Field Descriptions 930
11-7. Priority Status Register (PRIORITYSTAT) Field Descriptions 931
11-8. Mode Register (MODE) Field Descriptions 932
11-9. Control Register (CONTROL) Field Descriptions 934
11-10. Burst Size Register (BURST_SIZE) Field Descriptions 936
11-11. Burst Count Register (BURST_COUNT) Field Descriptions 936
11-12. Source Burst Step Size Register (SRC_BURST_STEP) Field Descriptions 937
11-13. Destination Burst Step Register Size (DST_BURST_STEP) Field Descriptions 938
11-14. Transfer Size Register (TRANSFER_SIZE) Field Descriptions 938
11-15. Transfer Count Register (TRANSFER_COUNT) Field Descriptions 939
11-16. Source Transfer Step Size Register (SRC_TRANSFER_STEP) Field Descriptions 939
11-17. Destination Transfer Step Size Register (DST_TRANSFER_STEP) Field Descriptions 940
11-18. Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE) Field Descriptions 940
11-19. Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT) Field Descriptions 941
11-20. Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP) Field Descriptions 941
11-21. Shadow Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW) Field Descriptions 942
11-22. Active Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR/DST_BEG_ADDR) Field Descriptions 942
11-23. Shadow Destination Begin and Current Address Pointer Registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW) Field Descriptions 943
11-24. Active Destination Begin and Current Address Pointer Registers (SRC_ADDR/DST_ADDR) Field Descriptions 943
12-1. SPI Module Signal Summary 948
12-2. SPI Registers 948
12-3. SPI Clocking Scheme Selection Guide 954
12-4. SPI Interrupt Flag Modes 957
12-5. 4-wire vs. 3-wire SPI Pin Functions 958
12-6. 3-Wire SPI Pin Configuration 959
12-7. Loopback Modes 962
12-8. SPI Configuration Control Register (SPICCR) Field Descriptions 964
12-9. Character Length Control Bit Values 965
12-10. SPI Operation Control Register (SPICTL) Field Descriptions 965
12-11. SPI Status Register (SPIST) Field Descriptions 966
12-12. Field Descriptions 967
12-13. SPI Emulation Buffer Register (SPIRXEMU) Field Descriptions 968
12-14. SPI Serial Receive Buffer Register (SPIRXBUF) Field Descriptions 968
12-15. SPI Serial Transmit Buffer Register (SPITXBUF) Field Descriptions 969
12-16. SPI Serial Data Register (SPIDAT) Field Descriptions 969
12-17. SPI FIFO Transmit (SPIFFTX) Register Field Descriptions 970
12-18. SPI FIFO Receive (SPIFFRX) Register Field Descriptions 971
12-19. SPI FIFO Control (SPIFFCT) Register Field Descriptions 971
12-20. SPI Priority Control Register (SPIPRI) Field Descriptions 972
13-1. SCI-A Registers 982
13-2. SCI-B Registers 982
13-3. SCI Module Signal Summary 983
13-4. Programming the Data Format Using SCICCR 984
13-5. Asynchronous Baud Register Values for Common SCI Bit Rates 990

Copyright © 2012–2019, Texas Instruments Incorporated

13-6. SCI Interrupt Flags 991
13-7. SCIA Registers 994
13-8. SCIB Registers 994
13-9. SCI Communication Control Register (SCICCR) Field Descriptions 995
13-10. SCI Control Register 1 (SCICTL1) Field Descriptions 996
13-11. Baud-Select Register Field Descriptions 998
13-12. SCI Control Register 2 (SCICTL2) Field Descriptions 999
13-13. SCI Receiver Status Register (SCIRXST) Field Descriptions 1000
13-14. SCI Receive Data Buffer Register (SCIRXBUF) Field Descriptions 1002
13-15. SCI FIFO Transmit (SCIFFTX) Register Field Descriptions 1002
13-16. SCI FIFO Receive (SCIFFRX) Register Field Descriptions 1003
13-17. SCI FIFO Control (SCIFFCT) Register Field Descriptions 1004
13-18. SCI Priority Control Register (SCIPRI) Field Descriptions 1006
14-1. Operating Modes of the I2C Module 1012
14-2. Ways to Generate a NACK Bit 1015
14-3. Descriptions of the Basic I2C Interrupt Requests 1017
14-4. I2C Module Registers 1019
14-5. I2C Mode Register (I2CMDR) Field Descriptions 1020
14-6. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR 1022
14-7. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR 1022
14-8. I2C Extended Mode Register (I2CEMDR) Field Descriptions 1023
14-9. I2C Interrupt Enable Register (I2CIER) Field Descriptions 1025
14-10. I2C Status Register (I2CSTR) Field Descriptions 1026
14-11. I2C Interrupt Source Register (I2CISRC) Field Descriptions 1028
14-12. I2C Prescaler Register (I2CPSC) Field Descriptions 1029
14-13. I2C Clock Low-Time Divider Register (I2CCLKL) Field Description 1030
14-14. I2C Clock High-Time Divider Register (I2CCLKH) Field Description 1030
14-15. Dependency of Delay d on the Divide-Down Value IPSC 1031
14-16. I2C Slave Address Register (I2CSAR) Field Descriptions 1031
14-17. I2C Own Address Register (I2COAR) Field Descriptions 1031
14-18. I2C Data Count Register (I2CCNT) Field Descriptions 1032
14-19. I2C Data Receive Register (I2CDRR) Field Descriptions 1032
14-20. I2C Data Transmit Register (I2CDXR) Field Descriptions 1033
14-21. I2C Transmit FIFO Register (I2CFFTX) Field Descriptions 1033
14-22. I2C Receive FIFO Register (I2CFFRX) Field Descriptions 1034
15-1. McBSP Interface Pins/Signals 1038
15-2. Register Bits That Determine the Number of Phases, Words, and Bits 1045
15-3. Interrupts and DMA Events Generated by a McBSP 1049
15-4. Effects of DLB and CLKSTP on Clock Modes 1051
15-5. Choosing an Input Clock for the Sample Rate Generator with the SCLKME and CLKSM Bits 1051
15-6. Polarity Options for the Input to the Sample Rate Generator 1052
15-7. Input Clock Selection for Sample Rate Generator 1055
15-8. Block - Channel Assignment 1064
15-9. 2-Partition Mode 1065
15-10. 8-Partition mode 1065
15-11. Receive Channel Assignment and Control With Eight Receive Partitions 1067
15-12. Transmit Channel Assignment and Control When Eight Transmit Partitions Are Used 1068
15-13. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits 1069
15-14. Bits Used to Enable and Configure the Clock Stop Mode 1072

Copyright © 2012–2019, Texas Instruments Incorporated

15-15. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme 1073
15-16. Bit Values Required to Configure the McBSP as a SPI Master 1076
15-17. Bit Values Required to Configure the McBSP as an SPI Slave 1077
15-18. Register Bits Used to Reset or Enable the McBSP Receiver Field Descriptions 1079
15-19. Reset State of Each McBSP Pin 1079
15-20. Register Bit Used to Enable/Disable the Digital Loopback Mode 1080
15-21. Receive Signals Connected to Transmit Signals in Digital Loopback Mode 1080
15-22. Register Bits Used to Enable/Disable the Clock Stop Mode 1080
15-23. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme 1081
15-24. Register Bit Used to Enable/Disable the Receive Multichannel Selection Mode 1081
15-25. Register Bit Used to Choose One or Two Phases for the Receive Frame 1081
15-26. Register Bits Used to Set the Receive Word Length(s) 1082
15-27. Register Bits Used to Set the Receive Frame Length 1082
15-28. How to Calculate the Length of the Receive Frame 1083
15-29. Register Bit Used to Enable/Disable the Receive Frame-Synchronization Ignore Function 1083
15-30. Register Bits Used to Set the Receive Companding Mode 1084
15-31. Register Bits Used to Set the Receive Data Delay 1085
15-32. Register Bits Used to Set the Receive Sign-Extension and Justification Mode 1087
15-33. Example: Use of RJUST Field With 12-Bit Data Value ABCh 1087
15-34. Example: Use of RJUST Field With 20-Bit Data Value ABCDEh 1087
15-35. Register Bits Used to Set the Receive Interrupt Mode 1088
15-36. Register Bits Used to Set the Receive Frame Synchronization Mode 1088
15-37. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin 1089
15-38. Register Bit Used to Set Receive Frame-Synchronization Polarity 1090
15-39. Register Bits Used to Set the SRG Frame-Synchronization Period and Pulse Width 1091
15-40. Register Bits Used to Set the Receive Clock Mode 1092
15-41. Receive Clock Signal Source Selection 1093
15-42. Register Bit Used to Set Receive Clock Polarity 1093
15-43. Register Bits Used to Set the Sample Rate Generator (SRG) Clock Divide-Down Value 1095
15-44. Register Bit Used to Set the SRG Clock Synchronization Mode 1095
15-45. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock) 1095
15-46. Register Bits Used to Set the SRG Input Clock Polarity 1097
15-47. Register Bits Used to Place Transmitter in Reset Field Descriptions 1098
15-48. Register Bit Used to Enable/Disable the Digital Loopback Mode 1099
15-49. Receive Signals Connected to Transmit Signals in Digital Loopback Mode 1099
15-50. Register Bits Used to Enable/Disable the Clock Stop Mode 1099
15-51. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme 1100
15-52. Register Bits Used to Enable/Disable Transmit Multichannel Selection 1101
15-53. Register Bit Used to Choose 1 or 2 Phases for the Transmit Frame 1101
15-54. Register Bits Used to Set the Transmit Word Length(s) 1101
15-55. Register Bits Used to Set the Transmit Frame Length 1103
15-56. How to Calculate Frame Length 1103
15-57. Register Bit Used to Enable/Disable the Transmit Frame-Synchronization Ignore Function 1104
15-58. Register Bits Used to Set the Transmit Companding Mode 1105
15-59. Register Bits Used to Set the Transmit Data Delay 1106
15-60. Register Bit Used to Set the Transmit DXENA (DX Delay Enabler) Mode 1108
15-61. Register Bits Used to Set the Transmit Interrupt Mode 1108
15-62. Register Bits Used to Set the Transmit Frame-Synchronization Mode 1109
15-63. How FSXM and FSGM Select the Source of Transmit Frame-Synchronization Pulses 1109

Copyright © 2012–2019, Texas Instruments Incorporated

15-64. Register Bit Used to Set Transmit Frame-Synchronization Polarity 1110
15-65. Register Bits Used to Set SRG Frame-Synchronization Period and Pulse Width 1111
15-66. Register Bit Used to Set the Transmit Clock Mode 1112
15-67. How the CLKXM Bit Selects the Transmit Clock and the Corresponding Status of the MCLKX pin 1112
15-68. Register Bit Used to Set Transmit Clock Polarity 1112
15-69. McBSP Emulation Modes Selectable with FREE and SOFT Bits of SPCR2 1114
15-70. Reset State of Each McBSP Pin 1114
15-71. McBSP Register Summary 1119
15-72. Serial Port Control 1 Register (SPCR1) Field Descriptions 1121
15-73. Serial Port Control 2 Register (SPCR2) Field Descriptions 1124
15-74. Receive Control Register 1 (RCR1) Field Descriptions 1126
15-75. Frame Length Formula for Receive Control 1 Register (RCR1) 1127
15-76. Receive Control Register 2 (RCR2) Field Descriptions 1127
15-77. Frame Length Formula for Receive Control 2 Register (RCR2) 1128
15-78. Transmit Control 1 Register (XCR1) Field Descriptions 1129
15-79. Frame Length Formula for Transmit Control 1 Register (XCR1) 1129
15-80. Transmit Control 2 Register (XCR2) Field Descriptions 1130
15-81. Frame Length Formula for Transmit Control 2 Register (XCR2) 1131
15-82. Sample Rate Generator 1 Register (SRGR1) Field Descriptions 1132
15-83. Sample Rate Generator 2 Register (SRGR2) Field Descriptions 1133
15-84. Multichannel Control 1 Register (MCR1) Field Descriptions 1134
15-85. Multichannel Control 2 Register (MCR2) Field Descriptions 1136
15-86. Pin Control Register (PCR) Field Descriptions 1138
15-87. Pin Configuration 1140
15-88. Receive Channel Enable Registers (RCERA...RCERH) Field Descriptions 1140
15-89. Use of the Receive Channel Enable Registers 1141
15-90. Transmit Channel Enable Registers (XCERA...XCERH) Field Descriptions 1142
15-91. Use of the Transmit Channel Enable Registers 1143
15-92. Receive Interrupt Sources and Signals 1144
15-93. Transmit Interrupt Sources and Signals 1145
15-94. Error Flags 1145
15-95. McBSP Interrupt Enable Register (MFFINT) Field Descriptions 1146
15-96. McBSP Mode Selection 1146
16-1. µDMA Channel Assignment Mapping 1151
16-2. Request Type Support 1153
16-3. Control Structure Memory Map 1154
16-4. Channel Control Structure 1154
16-5. µDMA Read Example: 8-Bit Peripheral 1163
16-6. µDMA Interrupt Assignments 1164
16-7. Channel Control Structure Offsets for Channel 30 1165
16-8. Channel Control Word Configuration for Memory Transfer Example 1165
16-9. Channel Control Structure Offsets for Channel 7 1166
16-10. Channel Control Word Configuration for Peripheral Transmit Example 1166
16-11. Primary and Alternate Channel Control Structure Offsets for Channel 8 1167
16-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example 1168
16-13. 1169
16-14. DMA Channel Source Address End Pointer (DMASRCENDP) Register Field Descriptions 1171
16-15. DMA Channel Destination Address End Pointer (DMADSTENDP) Register Field Descriptions 1172
16-16. DMA Channel Control Word (DMACHCTL) Register Field Descriptions 1172

Copyright © 2012–2019, Texas Instruments Incorporated

16-17. DMA Status (DMASTAT) Register Field Descriptions 1175
16-18. DMA Configuration (DMACFG) Register Field Descriptions 1176
16-19. DMA Channel Control Base Pointer (DMACTLBASE) Register Field Descriptions 1177
16-20. DMA Alternate Channel Control Base Pointer (DMAALTBASE) Register Field Descriptions 1177
16-21. DMA Channel Wait-on-Request Status (DMAWAITSTAT) Register Field Descriptions 1177
16-22. DMA Channel Software Request (DMASWREQ) Register Field Descriptions 1178
16-23. DMA Channel Useburst Set (DMAUSEBURSTSET) Register Field Descriptions 1178
16-24. DMA Channel Useburst Clear (DMAUSEBURSTCLR) Register Field Descriptions 1179
16-25. DMA Channel Request Mask Set (DMAREQMASKSET) Register Field Descriptions 1179
16-26. DMA Channel Request Mask Clear (DMAREQMASKCLR) Register Field Descriptions 1179
16-27. DMA Channel Enable Set (DMAENASET) Register Field Descriptions 1180
16-28. DMA Channel Enable Clear (DMAENACLR) Register Field Descriptions 1180
16-29. DMA Channel Primary Alternate Set (DMAALTSET) Register Field Descriptions 1180
16-30. DMA Channel Primary Alternate Clear (DMAALTCLR) Register Field Descriptions 1181
16-31. DMA Channel Priority Set (DMAPRIOSET) Register Field Descriptions 1181
16-32. DMA Channel Priority Clear (DMAPRIOCLR) Register Field Descriptions 1182
16-33. DMA Bus Error Clear (DMAERRCLR) Register Field Descriptions 1182
16-34. DMA Channel Assignment (DMACHALT) Register Field Descriptions 1182
16-35. DMA Channel Map Assignment (DMACHMAP0) Register Field Descriptions 1183
16-36. DMA Channel Map Assignment (DMACHMAP1) Register Field Descriptions 1184
16-37. DMA Channel Map Assignment (DMACHMAP2) Register Field Descriptions 1185
16-38. DMA Channel Map Assignment (DMACHMAP3) Register Field Descriptions 1186
16-39. DMA Peripheral Identification 1 (DMAPeriphID1) Register Field Descriptions 1187
16-40. DMA Peripheral Identification 2 (DMAPeriphID2) Register Field Descriptions 1187
16-41. DMA Peripheral Identification 3 (DMAPeriphID3) Register Field Descriptions 1188
16-42. DMA Peripheral Identification 4 (DMAPeriphID4) Register Field Descriptions 1188
16-43. DMA PrimeCell Identification 0 (DMAPCellID0) Register Field Descriptions 1188
16-44. DMA PrimeCell Identification 1 (DMAPCellID1) Register Field Descriptions 1189
16-45. DMA PrimeCell Identification 2 (DMAPCellID2) Register Field Descriptions 1189
16-46. DMA PrimeCell Identification 3 (DMAPCellID3) Register Field Descriptions 1189
17-1. EPI SDRAM Signal Connections 1196
17-2. CSCFGEXT + CSCFG Encodings 1200
17-3. Dual- and Quad- Chip Select Address Mappings 1201
17-4. Chip Select Configuration Register Assignment 1202
17-5. Capabilities of Host Bus 8 and Host Bus 16 Modes 1202
17-6. EPI Host-Bus 8 Signal Connections 1204
17-7. EPI Host-Bus 16 Signal Connections 1206
17-8. Data Phase Wait State Programming 1211
17-9. EPI General-Purpose Signal Connections 1218
17-10. Control Subsystem Address Mapping 1224
17-11. External Peripheral Interface (EPI) Register Map M3 Base Address: 0x400D_0000, C28x Base Address: 0x7C00 1225
17-12. C28x Base Address: 0x4430 1227
17-13. Base Address 0x400F_ B930 1227
17-14. EPI Configuration Register (EPICFG) Field Descriptions 1228
17-15. EPI Main Baud Rate (EPIBAUD) Register Field Descriptions 1229
17-16. EPI Main Baud Rate (EPIBAUD2) Register Field Descriptions 1230
17-17. EPI SDRAM Configuration (EPISDRAMCFG) Register Field Descriptions 1231
17-18. EPI Host-Bus 8 Configuration (EPIHB8CFG) Register Field Descriptions 1232

Copyright © 2012–2019, Texas Instruments Incorporated

17-19. EPI Host-Bus 16 Configuration (EPIHB16CFG) Register Field Descriptions 1235
17-20. EPI General-Purpose Configuration (EPIGPCFG) Register Field Descriptions..... 1238
17-21. EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2) Register Field Descriptions 1241
17-22. EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2) Register Field Descriptions 1244
17-23. EPI General-Purpose Configuration 2 (EPIGPCFG2) Register Field Descriptions..... 1247
17-24. EPI Address Map (EPIADDRMAP) Register Field Descriptions 1248
17-25. EPI Read Size 0 (EPIRSIZE0) Register and EPI Read Size 1 (EPIRSIZE1) Register Field Descriptions.. 1250
17-26. EPI Read Address 0 (EPIRADDR0) Register and EPI Read Address 1 (EPIRADDR1) Register Field Descriptions 1251
17-27. EPI Non-Blocking Read Data 0 (EPIRPSTD0) Register and EPI Non-Blocking Read Data 1 (EPIRPSTD1) Register Field Descriptions..... 1252
17-28. EPI Status (EPISTAT) Register Field Descriptions 1253
17-29. EPI Read FIFO Count (EPIRFIFOCNT) Register Field Descriptions 1254
17-30. EPI Read FIFO (EPIREADFIFO) Register and EPI Read FIFO Alias 1-7 (EPIREADFIFO1-7) Registers Field Descriptions 1255
17-31. EPI FIFO Level Selects (EPIFIFOLVL) Register Field Descriptions 1256
17-32. EPI Write FIFO Count (EPIWFIFOCNT) Register Field Descriptions..... 1257
17-33. EPI DMA Transmit Count (EPIDMATXCNT) Register Field Descriptions 1258
17-34. EPI Interrupt Mask (EPIIM) Register Field Descriptions 1258
17-35. EPI Raw Interrupt Status (EPIRIS) Register Field Descriptions 1259
17-36. EPI Masked Interrupt Status (EPIMIS) Register Field Descriptions..... 1260
17-37. EPI Error Interrupt Status and Clear (EPIEISC) Register Field Descriptions 1262
17-38. EPI Host-Bus 8 Configuration 3 Register (EPIHB8CFG3) Field Descriptions 1263
17-39. EPI Host-Bus 16 Configuration 3 Register (EPIHB16CFG3) Field Descriptions 1264
17-40. EPI Host-Bus 8 Configuration 4 Register (EPIHB8CFG4) Field Descriptions 1266
17-41. EPI Host-Bus 16 Configuration 4 Register (EPIHB16CFG4) Field Descriptions 1267
17-42. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME) Field Descriptions..... 1268
17-43. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME) Field Descriptions..... 1269
17-44. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME2) Field Descriptions 1270
17-45. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME2) Field Descriptions 1271
17-46. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME3) Field Descriptions 1272
17-47. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME3) Field Descriptions 1273
17-48. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME3) Field Descriptions 1274
17-49. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME4) Field Descriptions 1275
17-50. CEPIRTWCFG Register Field Descriptions 1276
17-51. CEPIRTWCNT Register Field Descriptions..... 1276
17-52. CEPIRTWPRD Register Field Descriptions 1277
17-53. CEPISTATUS Register Field Descriptions..... 1277
17-54. MEMPROT Register Field Descriptions..... 1278
18-1. Remainder (MAXLOAD/4)..... 1290
18-2. Actual Bytes Read 1290
18-3. Packet Sizes That Clear RXRDY 1290
18-4. Universal Serial Bus (USB) Controller Register Map 1292
18-5. Function Address Register (USBFADDR) Field Descriptions 1300
18-6. Power Management Register (USBPOWER) in OTG A/Host Mode Field Descriptions 1301
18-7. Power Management Register (USBPOWER) in OTG B/Device Mode Field Descriptions 1301
18-8. USB Transmit Interrupt Status Register (USBTXIS) Field Descriptions 1303
18-9. USB Receive Interrupt Status Register (USBRXIS) Field Descriptions..... 1305
18-10. USB Transmit Interrupt Status Register (USBTXIE) Field Descriptions 1307
18-11. USB Receive Interrupt Register (USBRXIE) Field Descriptions 1309

Copyright © 2012–2019, Texas Instruments Incorporated

18-12. USB General Interrupt Status Register (USBIS) in OTG A/Host Mode Field Descriptions 1311
18-13. USB General Interrupt Status Register (USBIS) in OTG B/Device Mode Field Descriptions 1312
18-14. USB Interrupt Enable Register (USBIE) in OTG A/Host Mode Field Descriptions 1313
18-15. USB Interrupt Enable Register (USBIE) in OTG B/Device Mode Field Descriptions 1314
18-16. Frame Number Register (FRAME) Field Descriptions 1315
18-17. USB Endpoint Index Register (USBEPIDX) Field Descriptions 1315
18-18. USB Test Mode Register (USBTEST) in OTG A/Host Mode Field Descriptions 1316
18-19. USB Test Mode Register (USBTEST) in OTG B/Device Mode Field Descriptions 1316
18-20. USB FIFO Endpoint *n* Register (USBFIFO[*n*]) Field Descriptions 1318
18-21. USB Device Control Register (USBDEVCTL) Field Descriptions 1319
18-22. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ) Field Descriptions 1321
18-23. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ) Field Descriptions 1322
18-24. USB Transmit FIFO Start Address Register (USBTXFIFOADDR) Field Descriptions 1323
18-25. USB Receive FIFO Start Address Register (USBRXFIFOADDR) Field Descriptions 1324
18-26. USB Connect Timing Register (USBCONTIM) Field Descriptions 1325
18-27. USB OTG VBUS Pulse Timing Register (USBVPLEN) Field Descriptions 1325
18-28. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF) Field Descriptions 1326
18-29. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF) Field Descriptions 1326
18-30. USB Transmit Functional Address Endpoint *n* Registers (USBTXFUNCADDR[*n*]) Field Descriptions 1327
18-31. USB Transmit Hub Address Endpoint *n* Registers(USBTXHUBADDR[*n*]) Field Descriptions 1328
18-32. USB Transmit Hub Port Endpoint *n* Registers(USBTXHUBPORT[*n*]) Field Descriptions 1329
18-33. USB Recieve Functional Address Endpoint *n* Registers(USBFIFO[*n*]) Field Descriptions 1330
18-34. USB Receive Hub Address Endpoint *n* Registers(USBRXHUBADDR[*n*]) Field Descriptions 1331
18-35. USB Transmit Hub Port Endpoint *n* Registers(USBRXHUBPORT[*n*]) Field Descriptions 1332
18-36. USB Maximum Transmit Data Endpoint *n* Registers(USBTXMAXP[*n*]) Field Descriptions 1333
18-37. USB Control and Status Endpoint 0 Low Register(USBCSRL0) in OTG A/Host Mode Field Descriptions . 1334
18-38. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in OTG B/Device Mode Field Descriptions 1335
18-39. USB Control and Status Endpoint 0 High Register (USBCSRH0) in OTG A/Host Mode Field Descriptions 1336
18-40. USB Control and Status Endpoint 0 High Register (USBCSRH0) in OTG B/Device Mode Field Descriptions 1336
18-41. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0) Field Descriptions 1337
18-42. USB Type Endpoint 0 Register (USBTYPE0) Field Descriptions 1337
18-43. USB NAK Limit Register (USBNAKLMT) Field Descriptions 1338
18-44. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in OTG A/Host Mode Field Descriptions 1339
18-45. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in OTG B/Device Mode Field Descriptions 1340
18-46. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in OTG A/Host Mode Field Descriptions 1342
18-47. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in OTG B/Device Mode Field Descriptions 1343
18-48. USB Maximum Receive Data Endpoint *n* Registers (USBTXMAXP[*n*]) Field Descriptions 1344
18-49. USB Control and Status Endpoint n Low Register(USBCSRL[*n*]) in OTG A/Host Mode Field Descriptions 1345
18-50. USB Control and Status Endpoint 0 Low Register(USBCSRL[*n*]) in OTG B/Device Mode Field Descriptions 1346
18-51. USB Control and Status Endpoint n High Register (USBCSRH[*n*]) in OTG A/Host Mode Field Descriptions 1348
18-52. USB Control and Status Endpoint 0 High Register(USBCSRH[*n*]) in OTG B/Device Mode Field Descriptions 1349

Copyright © 2012–2019, Texas Instruments Incorporated

18-53. USB Maximum Receive Data Endpoint *n* Registers (USBRXCOUNT[*n*]) Field Descriptions 1350
18-54. USB Host Transmit Configure Type Endpoint n Register(USBTXTYPE[*n*]) Field Descriptions 1351
18-55. USBTXINTERVAL[*n*] Frame Numbers 1352
18-56. USB Host Transmit Interval Endpoint n Register(USBTXINTERVAL[*n*]) Field Descriptions 1352
18-57. USB Host Configure Receive Type Endpoint n Register(USBRXTYPE[*n*]) Field Descriptions 1353
18-58. USBRXINTERVAL[*n*] Frame Numbers 1354
18-59. USB Host Receive Polling Interval Endpoint n Register(USBRXINTERVAL[*n*]) Field Descriptions 1354
18-60. USB Request Packet Count in Block Transfer Endpoint *n* Registers (USBRQPKTCOUNT[*n*]) Field Descriptions 1355
18-61. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS) Field Descriptions 1356
18-62. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS) Field Descriptions 1358
18-63. USB External Power Control Register (USBEPC) Field Descriptions 1360
18-64. USB External Power Control Raw Interrupt Status Register (USBEPCRIS) Field Descriptions 1362
18-65. USB External Power Control Interrupt Mask Register (USBEPCIM) Field Descriptions 1363
18-66. USB External Power Control Interrupt Status and Clear Register (USBEPCISC) Field Descriptions 1364
18-67. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions 1365
18-68. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions 1366
18-69. USB Device RESUME Interrupt Status and Clear Register (USBDRISC) Field Descriptions 1367
18-70. USB General-Purpose Control and Status Register (USBGPCS) Field Descriptions 1368
18-71. USB VBUS Droop Control Register (USBVDC) Field Descriptions 1369
18-72. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCRIS) Field Descriptions 1370
18-73. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCIM) Field Descriptions 1371
18-74. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCISC) Field Descriptions 1372
18-75. USB ID Valid Detect Raw Interrupt Status Register (USBIDVRIS) Field Descriptions 1373
18-76. USB ID Valid Detect Interrupt Mask Register (USBIDVIM) Field Descriptions 1374
18-77. USB ID Valid Detect Interrupt Status and Clear Register (USBIDVISC) Field Descriptions 1375
18-78. USB DMA Select Register (USBDMASEL) Field Descriptions 1376
19-1. TX & RX FIFO Organization 1384
19-2. Ethernet Register Map 1387
19-3. Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK) Register Field Descriptions 1388
19-4. Ethernet MAC Interrupt Mask (MACIM) Register Field Descriptions 1390
19-5. Ethernet MAC Receive Control (MACRCTL) Register Field Descriptions 1391
19-6. Ethernet MAC Transmit Control (MACTCTL) Register Field Descriptions 1391
19-7. Ethernet MAC Data (MACDATA) Register (READ) Field Descriptions 1393
19-8. Ethernet MAC Data (MACDATA) Register (WRITE) Field Descriptions 1393
19-9. Ethernet MAC Individual Address 0 (MACIA0) Register Field Descriptions 1394
19-10. Ethernet MAC Individual Address 0 (MACIA1) Register Field Descriptions 1395
19-11. Ethernet MAC Threshold (MACTHR) Register Field Descriptions 1396
19-12. Ethernet MAC Management Control (MACMCTL) Register Field Descriptions 1397
19-13. Ethernet MAC Management Divider (MACMDV) Register Field Descriptions 1398
19-14. Ethernet MAC Management Address Register (MACMAR) Field Descriptions 1398
19-15. Ethernet MAC Management Transmit Data (MACMTXD) Register Field Descriptions 1399
19-16. Ethernet MAC Management Receive Data (MACMRXD) Register Field Descriptions 1400
19-17. Ethernet MAC Number of Packets (MACNP) Register Field Descriptions 1400
19-18. Ethernet MAC Transmission Request (MACTR) Register Field Descriptions 1401
19-19. Ethernet MAC Timer Support (MACTS) Register Field Descriptions 1401
19-20. Ethernet PHY Management Register 0 – Control (MR0) Register Field Descriptions 1402
19-21. Ethernet PHY Management Register 1 – Control (MR1) Register Field Descriptions 1404
19-22. Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2) Register Field Descriptions 1405

Copyright © 2012–2019, Texas Instruments Incorporated

19-23. Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3) Register Field Descriptions 1405
19-24. Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4) Register Field Descriptions 1406
19-25. Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5) Register Field Descriptions 1407
19-26. Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6) Register Field Descriptions 1408
20-1. SSI Base Address Table (CM) 1419
20-2. SSI_REGS Registers 1420
20-3. SSI_REGS Access Type Codes 1420
20-4. SSICR0 Register Field Descriptions 1422
20-5. SSICR1 Register Field Descriptions 1424
20-6. SSIDR Register Field Descriptions 1426
20-7. SSISR Register Field Descriptions 1427
20-8. SSICPSR Register Field Descriptions 1428
20-9. SSIIM Register Field Descriptions 1429
20-10. SSIRIS Register Field Descriptions 1430
20-11. SSIMIS Register Field Descriptions 1432
20-12. SSIICR Register Field Descriptions 1434
20-13. SSIDMACTL Register Field Descriptions 1435
20-14. SSIPV Register Field Descriptions 1436
20-15. SSIPP Register Field Descriptions 1437
20-16. SSIPC Register Field Descriptions 1438
20-17. SSIPeriphID4 Register Field Descriptions 1439
20-18. SSIPeriphID5 Register Field Descriptions 1440
20-19. SSIPeriphID6 Register Field Descriptions 1441
20-20. SSIPeriphID7 Register Field Descriptions 1442
20-21. SSIPeriphID0 Register Field Descriptions 1443
20-22. SSIPeriphID1 Register Field Descriptions 1444
20-23. SSIPeriphID2 Register Field Descriptions 1445
20-24. SSIPeriphID3 Register Field Descriptions 1446
20-25. SSIPCellID0 Register Field Descriptions 1447
20-26. SSIPCellID1 Register Field Descriptions 1448
20-27. SSIPCellID2 Register Field Descriptions 1449
20-28. SSIPCellID3 Register Field Descriptions 1450
21-1. Register Map 1461
21-2. UART Data Register (UARTDR) Field Descriptions 1462
21-3. UART Receive Status Register (UARTRSR/UARTECR) Field Descriptions 1463
21-4. UART Error Clear (UARTECR) Register Field Descriptions 1464
21-5. UART Flag Register (UARTFR) Field Descriptions 1464
21-6. UART IrDA Low-Power Register (UARTILPR) Field Descriptions 1466
21-7. UART Integer Baud-Rate Divisor (UARTIBRD) Register Field Descriptions 1466
21-8. UART Fractional Baud-Rate Divisor (UARTFBRD) Register Field Descriptions 1467
21-9. UART Line Control Register (UARTLCRH) Field Descriptions 1467
21-10. UART Control (UARTCTL) Register Field Descriptions 1469
21-11. UART Interrupt FIFO Level Select (UARTIFLS) Register Field Descriptions 1470
21-12. UART Interrupt Mask (UARTIM) Register Field Descriptions 1471
21-13. UART Raw Interrupt Status (UARTRIS) Register Field Descriptions 1473
21-14. UART Masked Interrupt Status (UARTMIS) Register Field Descriptions 1475
21-15. UART Interrupt Clear (UARTICR) Register Field Descriptions 1477

Copyright © 2012–2019, Texas Instruments Incorporated

21-16. UART DMA Control (UARTDMACTL) Register Field Descriptions 1478
21-17. UART LIN Control (UARTLCTL) Register Field Descriptions 1478
21-18. UART LIN Snap Shot (UARTLSS) Register Field Descriptions 1479
21-19. UART LIN Timer (UARTLTIM) Register Field Descriptions 1479
21-20. UART Peripheral Identification 4 (UARTPeriphID4) Register Field Descriptions 1480
21-21. UART Peripheral Identification 5 (UARTPeriphID5) Register Field Descriptions 1480
21-22. UART Peripheral Identification 6 (UARTPeriphID6) Register Field Descriptions 1480
21-23. UART Peripheral Identification 7 (UARTPeriphID7) Register Field Descriptions 1481
21-24. UART Peripheral Identification 0 (UARTPeriphID0) Register Field Descriptions 1481
21-25. UART Peripheral Identification 1 (UARTPeriphID1) Register Field Descriptions 1481
21-26. UART Peripheral Identification 2 (UARTPeriphID2) Register Field Descriptions 1482
21-27. UART Peripheral Identification 3 (UARTPeriphID3) Register Field Descriptions 1482
21-28. UART PrimeCell Identification 0 (UARTPCellID0) Register Field Descriptions 1482
21-29. UART PrimeCell Identification 1 (UARTPCellID1) Register Field Descriptions 1483
21-30. UART PrimeCell Identification 2 (UARTPCellID2) Register Field Descriptions 1483
21-31. UART PrimeCell Identification 3 (UARTPCellID3) Register Field Descriptions 1483
22-1. Examples of I2C Master Timer Period versus Speed Mode 1489
22-2. Inter-Integrated Circuit (I2C) Interface Register Map 1498
22-3. I2C Master Slave Address (I2CMSA) Register Field Descriptions 1499
22-4. I2C Master Control/Status (I2CMCS) (Read-Only) Register Field Descriptions 1500
22-5. I2C Master Control/Status (I2CMCS) Write-Only Register Field Descriptions 1501
22-6. Write Field Decoding for I2CMCS[3:0] Field 1501
22-7. I2C Master Data (I2CMDR) Register Field Descriptions 1504
22-8. I2C Master Data (I2CMDR) Register Field Descriptions 1504
22-9. I2C Master Interrupt Mask (I2CMIMR) Register Field Descriptions 1505
22-10. I2C Master Raw Interrupt Status (I2CMRIS) Register Field Descriptions 1505
22-11. I2C Master Masked Interrupt Status (I2CMMIS) Register Field Descriptions 1506
22-12. I2C Master Interrupt Clear (I2CMICR) Register Field Descriptions 1506
22-13. I2C Master Configuration (I2CMCR) Register Field Descriptions 1507
22-14. I2C Slave Own Address (I2CSOAR) Register Field Descriptions 1508
22-15. I2C Slave Control/Status (I2CSCSR) Register Field Descriptions (Read-Only) 1508
22-16. I2C Slave Control/Status (I2CSCSR) Register Field Descriptions (Write-Only) 1509
22-17. I2C Slave Data (I2CSDR) Register Field Descriptions 1509
22-18. I2C Slave Interrupt Mask (I2CSIMR) Register Field Descriptions 1509
22-19. I2C Slave Raw Interrupt Status (I2CSRIS) Register Field Descriptions 1510
22-20. I2C Slave Masked Interrupt Status (I2CSMIS) Register Field Descriptions 1511
22-21. I2C Slave Interrupt Clear (I2CSICR) Register Field Descriptions 1511
23-1. Programmable Ranges Required by CAN Protocol 1531
23-2. Message Object Field Descriptions 1540
23-3. Message RAM Addressing in Debug Mode 1542
23-4. CAN Control Registers 1543
23-5. CAN Control Register (CAN CTL) Field Descriptions 1545
23-6. Error and Status Register Field Descriptions 1546
23-7. Error Counter Register Field Descriptions 1548
23-8. Bit Timing Register Field Descriptions 1548
23-9. Descriptions 1549
23-10. Test Register Field Descriptions 1550
23-11. Parity Error Code Register Field Descriptions 1551
23-12. Auto-Bus-On Time Register Field Descriptions 1552

Copyright © 2012–2019, Texas Instruments Incorporated

23-13. Transmission Request Register Field Descriptions 1552
23-14. New Data Registers Field Descriptions 1553
23-15. Interrupt Pending Registers Field Descriptions 1553
23-16. Message Valid Registers Field Descriptions 1554
23-17. Interrupt Multiplexer Registers Field Descriptions 1554
23-18. IF1 and IF2 Command Register Field Descriptions 1556
23-19. IF1 and IF2 Mask Registers Field Descriptions 1558
23-20. IF1 and IF2 Arbitration Registers Field Descriptions 1559
23-21. IF1 and IF2 Message Control Registers Field Descriptions 1560
23-22. IF3 Observation Register Field Descriptions 1562
23-23. IF1 and IF2 Mask Registers Field Descriptions 1563
23-24. IF3 Arbitration Register Field Descriptions 1564
23-25. IF3 Message Control Register Field Descriptions 1564
23-26. IF3 Update Control Register Field Descriptions 1566
24-1. Summary of Processor Mode, Privilege Level, and Stack Use 1570
24-2. Processor Register Map 1571
24-3. Cortex General-Purpose Registers 0-12 (R0-R12) Field Descriptions 1572
24-4. Cortex General-Purpose Registers 0-12 (R0-R12) Field Descriptions 1572
24-5. Link Register Field Descriptions 1573
24-6. Program Counter Register Field Descriptions 1573
24-7. PSR Register Combinations 1574
24-8. Program Status Register (PSR) Field Descriptions 1574
24-9. Priority Mask Register (PRIMASK) Field Descriptions 1576
24-10. Fault Mask Register (FAULTMASK) Field Descriptions 1577
24-11. Base Priority Mask Register Field Descriptions 1577
24-12. Control Register (CONTROL) Field Descriptions 1578
24-13. Memory Access Behavior 1579
24-14. SRAM Memory Bit-Banding Regions 1581
24-15. Peripheral Memory Bit-Banding Regions 1581
24-16. Exception Types Description 1586
24-17. Interrupts 1586
24-18. Exception Return Behavior 1591
24-19. Faults 1592
24-20. Fault Status and Fault Address Registers 1593
24-21. Cortex-M3 Instruction Summary 1595
25-1. Core Peripheral Register Regions 1599
25-2. Memory Attributes Summary 1601
25-3. TEX, S, C, and B Bit Field Encoding 1604
25-4. Cache Policy for Memory Attribute Encoding 1605
25-5. **AP Bit Field Encoding** 1605
25-6. Memory Region Attributes for Concerto Microcontrollers 1605
25-7. Peripherals Register Map 1606
25-8. SysTick Control and Status Register (STCTRL) Field Descriptions 1609
25-9. SysTick Reload Value Register (STRELOAD) Field Descriptions 1610
25-10. SysTick Current Value Register (STCURRENT) Field Descriptions 1610
25-11. Interrupt 0-31 Set Enable (EN0) Register Field Descriptions 1611
25-12. Interrupt 32-63 Set Enable 1 (EN1) Register Field Descriptions 1612
25-13. Interrupt 64-95 Set Enable 2 (EN2) Register Field Descriptions 1612
25-14. Interrupt 96-127 Set Enable 3 (EN3) Register Field Descriptions 1613

Copyright © 2012–2019, Texas Instruments Incorporated

25-15. Interrupt 128-133 Set Enable 4 (EN4) Register Field Descriptions 1613
25-16. Interrupt 0-31 Clear Enable (DIS0) Register Field Descriptions 1614
25-17. Interrupt 32-63 Clear Enable (DIS1) Register Field Descriptions 1614
25-18. Interrupt 64-95 Clear Enable (DIS2) Register Field Descriptions 1615
25-19. Interrupt 96-127 Clear Enable (DIS3) Register Field Descriptions 1616
25-20. Interrupt 128-133 Clear Enable (DIS4) Register Field Descriptions 1617
25-21. Interrupt 0-31 Set Pending (PEND0) Register Field Descriptions 1617
25-22. Interrupt 32-63 Set Pending (PEND1) Register Field Descriptions 1618
25-23. Interrupt 64-95 Set Pending (PEND2) Register Field Descriptions 1618
25-24. Interrupt 96-127 Set Pending (PEND3) Register Field Descriptions 1619
25-25. Interrupt 128-133 Set Pending (PEND4) Register Field Descriptions 1619
25-26. Interrupt 0-31 Interrupt Clear Pending (UNPEND0) Register Field Descriptions 1620
25-27. Interrupt 32-63 Clear Pending (UNPEND1) Register Field Descriptions 1620
25-28. Interrupt 64-95 Clear Pending (UNPEND2) Register Field Descriptions 1621
25-29. Interrupt 96-127 Clear Pending (UNPEND3) Register Field Descriptions 1622
25-30. Interrupt 128-133 Clear Pending (UNPEND4) Register Field Descriptions 1623
25-31. Interrupt 0-31 Active Bit (ACTIVE0) Register Field Descriptions 1623
25-32. Interrupt 32-54 Active Bit (ACTIVE1) Register Field Descriptions 1624
25-33. Interrupt 64-95 Active Bit (ACTIVE2) Register Field Descriptions 1624
25-34. Interrupt 96-127 Active Bit (ACTIVE3) Register Field Descriptions 1625
25-35. Interrupt 128-133 Active Bit (ACTIVE4) Register Field Descriptions 1625
25-36. Interrupt 0-133 Priority (PRI0-PRI33) Registers Field Descriptions 1626
25-37. Software Trigger Interrupt (SWTRIG) Register Field Descriptions 1627
25-38. Auxiliary Control (ACTLR) Register Field Descriptions 1628
25-39. CPU ID Base (CPUID) Register Field Descriptions 1629
25-40. Interrupt Control and State (INTCTRL) Register Field Descriptions 1630
25-41. Vector Table Offset (VTABLE) Field Descriptions 1633
25-42. Interrupt Priority Levels 1634
25-43. Application Interrupt and Reset Control (APINT) Register Field Descriptions 1634
25-44. System Control (SYSCTRL) Register Field Descriptions 1636
25-45. Configuration and Control (CFGCTRL) Register Field Descriptions 1637
25-46. System Handler Priority 1 (SYSPRI1) Register Field Descriptions 1638
25-47. System Handler Priority 2 (SYSPRI2) Register Field Descriptions 1639
25-48. System Handler Priority 3 (SYSPRI3) Register Field Descriptions 1639
25-49. System Handler Control and State (SYSHNDCTRL) Register Field Descriptions 1640
25-50. Configurable Fault Status (FAULTSTAT) Register Field Descriptions 1643
25-51. Hard Fault Status (HFAULTSTAT) Register Field Descriptions 1647
25-52. Memory Management Fault Address (MMADDR) Register Field Descriptions 1648
25-53. Bus Fault Address (FAULTADDR) Register Field Descriptions 1648
25-54. MPU Type (MPUTYPE) Register Field Descriptions 1649
25-55. MPU Control (MPUCTRL) Register Field Descriptions 1650
25-56. MPU Region Number (MPUNUMBER) Register Field Descriptions 1651
25-57. MPU Region Base Address (MPUBASE) Register Field Descriptions 1652
25-58. Example SIZE Field Values 1652
25-59. MPU Region Attribute and Size (MPUATTR) Field Descriptions 1653

Copyright © 2012–2019, Texas Instruments Incorporated

Read This First

About This Manual

This Technical Reference Manual (TRM) details the integration, the environment, the functional description, and the programming models for each peripheral and subsystem in the device.

The TRM should not be considered a substitute for the data manual, rather a companion guide that should be used alongside the device-specific data manual to understand the details to program the device. The primary purpose of the TRM is to abstract the programming details of the device from the data manual. This allows the data manual to outline the high-level features of the device without unnecessary information about register descriptions or programming models.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers may be shown with the suffix h or the prefix 0x. For example, the following number is 40 hexadecimal (decimal 64): 40h or 0x40.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties with default reset value below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure can have one of multiple meanings:
    - Not implemented on the device
    - Reserved for future device expansion
    - Reserved for TI testing
    - Reserved configurations of the device that are not supported
  - Writing nondefault values to the Reserved bits could cause unexpected behavior and should be avoided.

Glossary

TI Glossary —This glossary lists and explains terms, acronyms, and definitions.

Related Documentation From Texas Instruments

For a complete listing of related documentation and development-support tools for these devices, visit the Texas Instruments website at http://www.ti.com. Additionally, the *TMS320C28x CPU and Instruction Set Reference Guide* (SPRU430) and *TMS320C28x Floating Point Unit and Instruction Set Reference Guide* (SPRUEO2) must be used in conjunction with this TRM.

(1)(2)(3)(4)(5)

(1) Concerto, Code Composer Studio, controlSUITE are trademarks of Texas Instruments.
(2) Cortex is a trademark of ARM Limited.
(3) ARM, Thumb, AMBA, ARMA are registered trademarks of ARM Limited.
(4) XDS is a trademark of XDS.
(5) is a trademark of ~ Texas Instruments.

Copyright © 2012–2019, Texas Instruments Incorporated

System Control and Interrupts

This chapter explains system control and interrupts for this device. The system control module configures and manages the overall operation of the device and provides information about the device operation. Configurable features in system control include reset control, NMI operation, power control, clock control, and low-power modes.

For more information on the Viterbi, Complex Math, CRC Unit (VCU), please refer to *TMS320C28x Extended Instruction Sets Technical Reference Manual*.

Topic **Page**

1.1	**Signal Description**	**80**
1.2	**System Control Functional Description**	**81**
1.3	**Reset Control**	**82**
1.4	**WIR Mode**	**92**
1.5	**Exceptions and Interrupts Control**	**95**
1.6	**Safety Features**	**119**
1.7	**Power Control**	**125**
1.8	**Clock Control**	**125**
1.9	**Low Power Modes**	**140**
1.10	**Code Security Module (CSM)**	**144**
1.11	**µCRC Module**	**155**
1.12	**Inter Processor Communications (IPC)**	**156**
1.13	**System Control Registers**	**165**

Copyright © 2012–2019, Texas Instruments Incorporated

1.1 Signal Description

Table 1-1 lists the external signals of the system control module and describes the function of each.

The NMI signal is one of the peripheral functions for the PB7_GPIO15 signal, and functions as a GPIO after reset. PB7_GPIO15 is under commit protection and requires a special process to be configured as any alternate function or to subsequently return to the GPIO function. The Pin Mux/ Pin Assignment column in Table 1-1 lists the GPIO options for the NMI signal. When the PJ7_GPIO63 pin is configured, it can be used as an XCLKIN pin to provide a clock source to the CAN and USB modules on this device. The PF2_GPIO34 pin can be used as XCLKOUT to monitor the PLL output clock when configured for either the master subsystem or control subsystem as shown in Table 1-1. For more information on configuring GPIOs, see the *General-Purpose Input/Outputs (GPIOs)* chapter.

The remaining signals (fixed) have a fixed pin assignment and function. Please refer to the device data manual for more details on the pin numbers.

Table 1-1. Signals for System Control and Clocks

Pin Functional Name	Pin Name(refer to datasheet for pin numbers)	Pin Mux/ Pin Assignment			Description
		Peripheral Mode	Alternate Mode Select	Core Select	
NMI	PB7_GPIO15	4	0 (default)	Master (default)	Non-maskable interrupt
XCLKIN	PJ7_GPIO63	0 (default)	0 (default)	Master (default)	External oscillator input. This pin feeds a clock from an external 3.3-V oscillator to the internal USB PLL module and to the CAN peripherals.
XCLKOUT	PF2_GPIO34	0 Don't Care	15 3	Master Control	External oscillator output. This pin outputs a clock divided-down from the internal PLL System Clock. The divide ratio is defined in the XPLLCLKCFG register.
X1	Refer to the data manual for pin no.	Fixed	Fixed	Fixed	On-chip crystal-oscillator input. To use this oscillator, a quartz crystal or a ceramic resonator must be connected across X1 and X2. In this case, the XCLKIN path must be disabled by bit 13 in the CLKCTL register. If this pin is not used, it must be tied to GND.
X2	Refer to the data manual for pin no.	Fixed	Fixed	Fixed	On-chip crystal-oscillator output. A quartz crystal or a ceramic resonator must be connected across X1 and X2. If X2 is not used, it must be left unconnected.
$\overline{\text{XRS}}$	Refer to the data manual for pin no.	Fixed	Fixed	Fixed	Digital Subsystem Reset (in) and Watchdog/Power-on Reset (out). In most applications, it is recommended that the $\overline{\text{XRS}}$ pin be tied with the $\overline{\text{ARS}}$ pin.
$\overline{\text{ARS}}$	Refer to the data manual for pin no.	Fixed	Fixed	Fixed	Analog subsystem Reset (in) and Power-on Reset (out). In most applications, it is recommended that the $\overline{\text{ARS}}$ pin be tied with the $\overline{\text{XRS}}$ pin.
$\overline{\text{TRST}}$	Refer to the data manual for pin no.	Fixed	Fixed	Fixed	JTAG test reset with internal pulldown
$\overline{\text{VREG18EN}}$	Refer to the data manual for pin no.	Fixed	Fixed	Fixed	Internal 1.8-V VREG Enable/Disable for VDD18. Pull low to enable the internal 1.8-V voltage regulator (VREG18), pull high to disable VREG18.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-1. Signals for System Control and Clocks (continued)

Pin Functional Name	Pin Name(refer to datasheet for pin numbers)	Pin Mux/ Pin Assignment			Description
		Peripheral Mode	Alternate Mode Select	Core Select	
$\overline{\text{VREG12EN}}$	Refer to the data manual for pin no.	Fixed	Fixed	Fixed	Internal 1.2-V VREG Enable/Disable for VDD12. Pull low to enable the internal 1.2-V voltage regulator (VREG12), pull high to disable VREG12.
Vssosc	Refer to the data manual for pin no.	Fixed	Fixed	Fixed	Clock Oscillator Ground Pin
EMU0	Refer to the data manual for pin no.	Fixed	Fixed	Fixed	Emulator pin 0. When TRST is driven high, this pin is used as an interrupt to or from the emulator system and is defined as input/output through the JTAG scan.
EMU1	Refer to the data manual for pin no.	Fixed	Fixed	Fixed	Emulator pin 1. When TRST is driven high, this pin is used as an interrupt to or from the emulator system and is defined as input/output through the JTAG scan.

1.2 System Control Functional Description

The system control module provides the following capabilities:

- Device identification and configuration registers
- Reset control
- Device WIR mode control
- Exceptions and Interrupt control
- Safety and error handling features of the device
- Power control
- Clock control
- Low Power modes
- Security module
- Inter-Processor Communication (IPC)

1.2.1 Device Identification

Device identification registers provide information on device class, device family, revision, part number, pin count, operating temperature range, package type, pin count and device qualification status.

All of the information is readable by master subsystem applications, and part number, part type and revision information is readable by both the master subsystem and control subsystem. Please refer to the *System Control Registers* section for more details on these registers.

1.2.1.1 Master Subsystem and Control Subsystem Device Identification

The master subsystem device identification registers are: DID0 and DID1. The control subsystem device identification registers are: PARTID and REVID.

Copyright © 2012–2019, Texas Instruments Incorporated

1.2.2 Device Configuration Registers

Several registers provide users with configuration information for debug/identification purposes on this device. This information provides the peripheral's features and indicates how much RAM and FLASH memory are available on this part. Both master and control subsystems have their own device configuration registers. The master subsystem can access the device configuration of both the master and control subsystems, whereas the control subsystem can only access its own device configuration registers. All device configuration registers are read-only in both systems.

1.2.2.1 Master Subsystem Device Configuration

Table 1-2 provides a description of the master subsystem device configuration.

Table 1-2. Master Subsystem Device Configuration

Name	Registers	Description
Device configuration or capabilities registers	DC1, DC2, DC4, DC6, DC7, DC10	If a particular bit in these registers is set to "1" then the associated feature or module is available in the device.
Shared memory configuration register	MEMCNF	If a particular bit is set to "1" then that particular shared memory is enabled.
M3 configuration register	MCNF	Flash bank size and uCRC configuration

1.2.2.2 Control Subsystem Device Configuration

The control subsystem core configuration registers are: CCNF0, CCNF1, CCNF2, CCNF3, and CCNF4. All these registers are defined in the Device Identification and Device Configuration registers subsection in the *System Control Registers* section.

1.3 Reset Control

This section describes the reset sources for the device and hardware functions during reset. How system software should handle the reset cause and what Boot ROM does to handle the reset cause is also discussed.

This device has two external reset pins: $\overline{\text{XRS}}$ for the digital subsystem and $\overline{\text{ARS}}$ for the analog subsystem of the chip. It is recommended that these two pins be externally tied together with a board signal trace. The $\overline{\text{XRS}}$ pin responds to an external reset signal and an internal power-on signal to reset the entire chip. The $\overline{\text{XRS}}$ pin also drives the reset signal out of the chip to external circuitry when the master watchdog timers (WDT0, WDT1) and master NMI watchdog timer (MNMIWD) expires. The $\overline{\text{XRS}}$ signal is internally gated with other internal reset signals to drive individual resets to the master subsystem, control subsystem, analog subsystem and the shared resources.

For all reset causes which reset the master subsystem, both the control subsystem and analog subsystem are also reset, and are held in reset until software running on the master subsystem brings them out of reset. The master subsystem boot ROM software, which gets executed each time the device is reset, will bring both the control and analog subsystems out of reset by setting the respective bits in the CRESCNF register.

1.3.1 Device Level Reset Sources

Table 1-3 shows various reset signals in this device and how it affects different hardware modules in the device.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-3. Device Level Reset Sources

No.	Reset Source	M3 Core Reset	C28 Core Reset	JTAG / Debug Logic Reset	Master subsystem Reset	Control subsystem Reset	Analog subsystem Reset	Flash Pump	IOs	XRSn output
1	POR reset	Yes	Yes	Yes	Yes	Yes	Yes	Yes	HiZ	Yes
2	$\overline{\text{XRS}}$ input	Yes	Yes	Yes	Yes	Yes	Yes	Yes	HiZ	---
3	WDT0 reset	Yes	Yes	No	Yes	Yes	Yes	No	HiZ	Yes
4	WDT1 reset	Yes	Yes	No	Yes	Yes	Yes	No	HiZ	Yes
5	MNMIWD	Yes	Yes	No	Yes	Yes	Yes	No	HiZ	Yes
6	M3 software reset / debugger reset	Yes	Yes	No	Yes	Yes	No	No	HiZ	No
7	C28 software RSn from M3(CRESCNF[M3RSnIN])	No	Yes	No	No	Yes	No	No	C28 GPIOs in input state	No
8	$\overline{\text{TRST}}$	No	No	Yes	No	No	No	No	No	No
9	C28 Debugger Reset ($\overline{\text{SYSRS}}$)	No	Yes	No	No	Yes	No	No	C28 GPIOs in input state	No
10	CNMIWD Reset	No	Yes	No	No	Yes	No	No	C28 GPIOs in input state	No
11	ACIB Reset	No	No	No	No	No	Yes	No	Analog IOs in GPIO input state	No

As shown in Table 1-3, because the Cortex-M3 core belongs to the master subsystem, whenever it is reset, the entire device is reset, including control and analog subsystems. Software on the master subsystem can choose to reset the control subsystem and analog subsystem by writing a "1" to M3RsnIN (bit 16) and bit $\overline{\text{ACIBRESET}}$ (bit 17) of the CRESCNF register.

Also note that whenever the ACIB reset is activated, the analog subsystem is reset. Whenever the analog subsystem is reset, the ACIB is also reset and vice-versa. On power-up or after every master subsystem reset, the analog subsystem is held in reset. This also means that ACIB is held in reset.

After a reset, the reset cause register (RESC) is updated with the reset cause. The bits in the reset cause register are sticky and maintain their state across multiple resets except when a Power-on Reset (POR) is the reset cause, in which case all the bits in the RESC register are cleared. For the master subsystem on this device, this register is called the MRESC register.

After any reset that resets the master core, boot ROM code on the master subsystem will run, which brings the control system and ACIB out of reset and starts the application software as per the boot mode configured. Please refer to the *Boot ROM* chapter for more details.

Table 1-4 provides information for the device bring up time-line on power-up.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-4. Device Bring-Up Time Line

Time Line	Master Subsystem	Control Subsystem	Analog Subsystem/ACIB
T0	POR indicates that device is not yet fully powered and is in reset	POR indicates that device is not yet fully powered and is in reset	POR indicates that device is not yet fully powered and is in reset
T1	Master subsystem out of reset. Device Specific initialization as mentioned in the Boot ROM guide.	Control subsystem held in reset	Analog subsystem/ACIB held in reset
T2	Bring control subsystem out of reset	Control subsystem starts executing C-Boot ROM	Analog subsystem/ACIB held in reset
T3	Bring ACIB out of reset		
T4	Master bootROM boots the device based on the boot mode GPIO configuration.	After system initialization, the C28x CPU goes to IDLE mode; wakes up on IPC interrupts	Analog subsystem/ACIB out of reset and analog peripherals are accessible for both master and control subsystems.
T5	Running application on master	Master Application decides how to run application on control subsystem.	ACIB and analog peripherals ready to be configured, controlled and monitored by control subsystem application and master subsystem application.

Below are the details on each reset source.

1.3.1.1 Power-on Reset (POR):

The Analog and Digital Subsystems' each have a Power-On-Reset (POR) circuit that creates a clean reset throughout the device enabling glitchless GPIOs during the power-on procedure. The POR function keeps both $\overline{\text{ARS}}$ and $\overline{\text{XRS}}$ driven low during device power up. Whenever the device is reset by a POR condition, bit 1 of the MRESC register is set indicating that the POR has caused a reset, and because a POR reset pulls the $\overline{\text{XRS}}$ pin low, bit 0 is also set.

NOTE: The power-on reset also resets the JTAG controller.

While in most applications, the POR generated reset has a long enough duration to also reset other system ICs, some applications may require a longer lasting pulse. In these cases, the ARS and XRS reset pins (which are open-drain) can also be driven low to match the time the device is held in a reset state with the rest of the system.

When the device is reset by a POR condition, the master subsystem begins executing the master Boot ROM and brings the control subsystem and analog subsystem out of reset. Once out of reset, the control subsystem starts executing its code in C-Boot ROM. Please refer to the *Boot ROM* chapter for how both the master subsystem boot ROM and control subsystem boot ROM handle this reset cause.

1.3.1.2 External Reset Input

There are two external reset input pins on the device: $\overline{\text{XRS}}$ and $\overline{\text{ARS}}$.

The $\overline{\text{XRS}}$ and $\overline{\text{ARS}}$ pins both respond to an external reset signal (active low), and can also drive the reset signal out of the chip to external circuitry whenever any of the internal digital or analog subsystem resets become active, as shown in the Table 1-3. The internal resets which pull $\overline{\text{XRS}}$ low are caused by the master subsystem watchdog modules (WDT0, WDT1 and MNMIWD). Whenever the master subsystem is reset by an $\overline{\text{XRS}}$ condition, it will hold both control and analog subsystems in reset and bit 0 of the MRESC register will be set to indicate an $\overline{\text{XRS}}$ caused the reset. When the device is out of reset, the master subsystem Boot ROM will start executing and it will bring both control and analog systems out of reset. Once out of reset, the control subsystem will start executing its own Boot ROM. Please refer to the *Boot ROM* chapter for how both the master and control boot ROMs handle an $\overline{\text{XRS}}$ reset.

It is recommended to connect both $\overline{\text{XRS}}$ and $\overline{\text{ARS}}$ pins together externally using a single trace.

The internal reset which can pull $\overline{\text{ARS}}$ low is caused by the power-on reset (POR).

Copyright © 2012–2019, Texas Instruments Incorporated

1.3.1.3 $\overline{\text{TRST}}$

$\overline{\text{TRST}}$ is a JTAG test reset with an internal pulldown, and when driven high, gives the scan system control of the operations of the device. If this signal is not connected or driven low, the device operates in functional mode, and the test reset signals are ignored. $\overline{\text{TRST}}$ resets the JTAG TAPs of both the master subsystem CPU and control subsystem CPU.

NOTE: TRST is an active-high test pin and must be maintained low during normal device operation.

The $\overline{\text{TRST}}$ reset does not show up on $\overline{\text{XRS}}$. Note that $\overline{\text{TRST}}$ is different from a debugger reset. The debugger reset on the Cortex-M3 is realized by writing to the NVIC registers and on the C28x CPU it is realized by writing to TI-internal hidden registers.

1.3.1.4 Watchdog Timer 0,1 Reset

This device has two watchdog timer modules on the master subsystem in case one watchdog clock source fails. Watchdog Timer 0 is run off the system clock and Watchdog Timer 1 is run off the main oscillator. Each module operates in the same manner except that because the Watchdog Timer 1 module is in a different clock domain, register accesses must have a time delay between them. The watchdog timer can be configured to generate an interrupt to the microcontroller on its first time-out and to generate a reset on its second time-out.

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

1. The watchdog timer times out for the second time without being serviced.
2. An internal reset is asserted which pulls $\overline{\text{XRS}}$ low and this resets the whole device and appropriate bits in the MRESC register are set.

Note that on this device, the watchdogs are available only on the master subsystem. The control subsystem does not have its own watchdog, but when the master subsystem watchdogs generate a reset, it will reset the entire device.

When the device is out of reset, the master subsystem boot ROM will start executing and it will bring both the control and analog subsystems out of reset. Once out of reset, the control subsystem will start executing its own boot ROM. Refer to the *Boot ROM* chapter for more details on how the master subsystem and control subsystem boot ROM handles watchdog timer resets.

1.3.1.5 Master NMI Watchdog Timer Reset

This device has a watchdog timer associated with its NMI logic, which when an enabled non-maskable interrupt is generated to the CPU, the NMIWD timer starts counting. The NMIWD will then generate a reset if the MNMIWDCNT counter register value reaches the value programmed in the MNMIWDPRD period register. Once the MNMIWD counter starts counting, it will stop and reset to 0 only if all the flags in the MNMIFLG register are cleared.

Note that the ACIBERR NMI is disabled on reset, and if the application software does not enable this NMI by setting bit 9 in the MNMICFG register, no NMI will be triggered to the CPU if these error conditions occur. Any other NMI condition will trigger an NMI to the CPU and the NMIWD counter will start counting.

Refer to the MNMI section in this document for more details on MNMI sources, and refer to the *Boot ROM* chapter for more details on how the master boot ROM software handles different NMIs.

Whenever an MNMIWD reset is generated, it will pull the $\overline{\text{XRS}}$ pin low. This will cause the entire device to reset and both the analog and control subsystems will be held in reset. Once out of reset, the master subsystem will start executing boot ROM and bring the analog and control subsystems out of reset. Once out of reset, the control subsystem will start executing its boot ROM. Refer to the *Boot ROM* chapter for more details on how boot ROM handles this reset.

Copyright © 2012–2019, Texas Instruments Incorporated

1.3.1.6 Master Software Reset and Master Debugger Reset

The master software reset is a software-generated reset output by the NVIC (using SYSRESETREQ/VECTRESET of the Cortex-M3 NVIC Application Interrupt and Reset Control Register). The master debugger reset is a debugger-generated reset that is also an output of the NVIC.

In addition to resetting the master subsystem, these two resets can propagate to the control subsystem.

Note that while debugging or developing code when the debug probe is connected to the device, it has to be in RUN state for the control subsystem to see the software reset and/or debugger reset from the master subsystem. If the control subsystem is in DEBUG HALT mode, it will not see this reset input.

As shown in Table 1-3, this reset does not pull the $\overline{\text{XRS}}$ signal low, but if both the cores are reset, when coming out of reset, each core will start executing its own boot ROM. On the master subsystem, bit 4 of the MRESC register will be set to inform the application software that the device was reset by software.

1.3.1.7 Control Subsystem Software Reset (from Master)

The control subsystem can be reset by software running on the master subsystem by writing to the M3RSnIN bit (bit 16) of the CRESCNF register. When the control subsystem is reset by the master subsystem by writing to the CRESCNF register, all the GPIOs which were previously owned by the control subsystem CPU will still be owned by the control subsystem CPU, and all the GPIOs will be configured as inputs.

The control subsystem will start executing C-Boot ROM when out of reset.

1.3.1.8 Control Subsystem Debugger Reset

This reset is driven by the debugger only to the control subsystem. As shown in Table 1-3, a control system debugger reset, resets the C28x CPU core and control subsystem. All the GPIOs which were configured for the control subsystem will remain with the control subsystem and will be reset to their default state (input, GPIO mode). Please refer to *GPIO* chapter of this document for more details on the reset state of the control subsystem GPIOs.

The control subsystem will start executing C-Boot ROM when out of reset.

1.3.1.9 Control Subsystem NMIWD Reset

This device has no general-purpose watchdog timer associated with the control subsystem but there is an NMI watchdog timer associated with the control subsystem's NMI logic. This timer will start counting as soon as an enabled non-maskable interrupt is triggered to the control subsystem CPU. It will reset the subsystem if the triggered NMI is not acknowledged, by clearing the respective bits in CNMIFLG register.

All of the GPIOs configured for the control subsystem will still be owned by the control subsystem and put to their default reset state after a control subsystem NMIWD reset. The CNMIWDRST bit (bit 16) of the CRESSTS register will be set for the master subsystem to know when the control subsystem is reset by the CNMIWD timer. Also, an NMI is triggered to the master subsystem indicating that the control subsystem was reset by the NMIWD.

Note that the master subsystem is not reset when the control subsystem is reset by the CNMIWD, but if the master subsystem is reset by a MNMIWD it will also reset the control subsystem. Refer to the CNMI and MNMI sections for more details on NMI logic.

1.3.1.10 ACIB Reset

The analog common interface bus reset resets the analog subsystem and analog common interface bus (ACIB). This reset signal can only be generated by software running on the master subsystem by setting the ACIBRST bit in the CRESCNF register.

This reset puts all the analog subsystem GPIOs in input mode. The master subsystem and the control subsystem are not reset by this signal, and the control subsystem can learn about the status of this signal by looking at the $\overline{\text{ACIBRESET}}$ bit (bit 30) of the DEVICECNF register. Refer to the DEVICECNF register for details.

Copyright © 2012–2019, Texas Instruments Incorporated

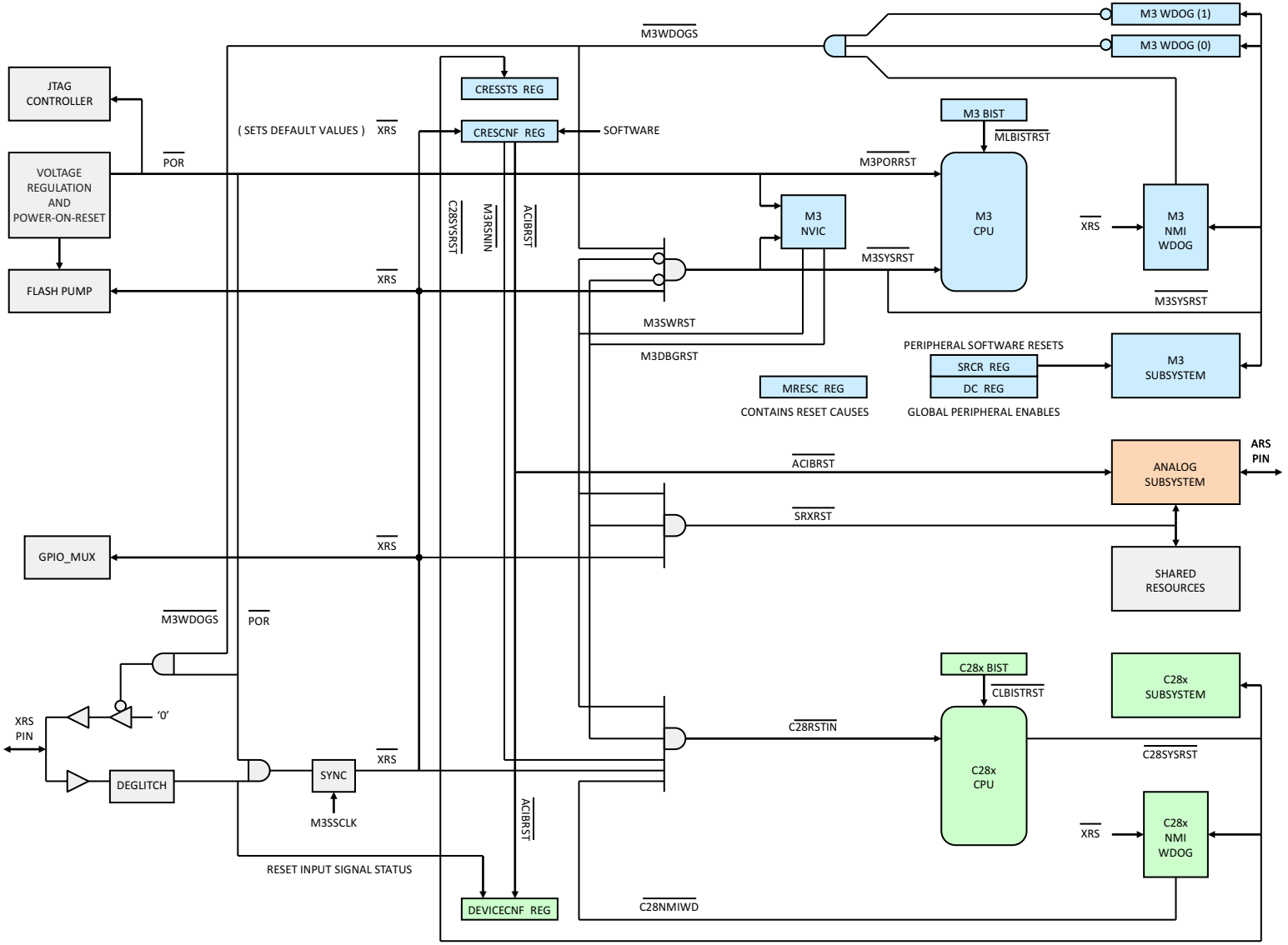
1.3.1.11 Shared Resources Reset

Shared resources between the master and control subsystems, (IPC registers, MTOC and CTOM message RAM) configurable shared RAMs, and the ACIB interface, are reset by the shared reset signal defined as $\overline{\text{SRXRST}}$. $\overline{\text{SRXRST}}$ is triggered by a master software reset or by a master debugger reset. $\overline{\text{SRXRST}}$ is also triggered by an $\overline{\text{XRS}}$.

Figure 1-1 shows reset connectivity on the device.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-1. Resets Connectivity



Copyright © 2012–2019, Texas Instruments Incorporated

1.3.2 Handling of Resets at System Level

This section explains resets at the subsystem level. It discusses how the master subsystem boot ROM and the control subsystem boot ROM handles the reset causes after a reset.

1.3.2.1 Master Subsystem Reset Handling

The master subsystem, the Cortex-M3 CPU, and the Nested Vectored Interrupt Controller (NVIC) are all reset by the Power-On Reset (POR) or the M3SYSRST reset signal. In both cases, the Cortex-M3 CPU restarts program execution from the address provided by the reset entry in the vector table.

A register can later be referenced to determine the source of the reset. The M3SYSRST signal also propagates to the Cortex-M3 peripherals and the rest of the Cortex-M3 subsystem. As shown in Figure 1-1, the M3SYSRST bit has four possible sources: XRS, M3WDOGS, M3SWRST, and M3DBGRST. The M3WDOGS is set in response to time-out conditions of the two Cortex-M3 watchdogs or the Cortex-M3 NMI Watchdog. The M3SWRST is a software-generated reset output by the NVIC. The M3DBGRST is a debugger-generated reset that is also output by the NVIC. In addition to driving M3SYSRST, these two resets also propagate to the control subsystem and the analog subsystem. The M3RSnIN bit can be set in the CRESCNF register to selectively reset the control subsystem from the master subsystem, and the ACIBRST bit of the same register selectively resets the analog common interface bus. In addition to driving reset signals to other parts of the chip, the master subsystem can also detect a C28SYSRST reset being set in the control subsystem by reading the CRES bit of the CRESSTS register.

The master subsystem can also set bits in the SRCR register to selectively reset individual Cortex-M3 peripherals, provided they are enabled inside the respective device configuration (DC) register.

For all the reset causes shown in Table 1-5, except for master software reset and master debugger reset, both analog and control subsystems are also reset and held in reset. The master subsystem will be first out of reset and starts executing software in master subsystem boot ROM. The master boot ROM brings the control subsystem and analog subsystem out of reset. The control subsystem starts executing software in the control subsystem boot ROM after it is out of reset. Refer to Section 1.3.2.1.1 and Section 1.3.2.1.2 for more details on how each boot ROM handles each reset cause.

Table 1-5 provides summary of results of various reset operations on the master subsystem.

Table 1-5. Master Subsystem Rests, Signals and Effects

No.	Reset Source	Device Reset Signals Asserted	Cortex-M3 CPU Core Reset	JTAG Reset	Master Subsystem Reset	Control Subsystem Reset	Analog Subsystem Reset	Shared Resources Reset	ACIB reset
1	POR	POR, XRS, M3SYSRST, M3PORRST, ACIBRST, SRXRST, C28RSTIN, C28SYSRTS	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	XRS	XRS, M3SYSRST, ACIBRST, SRXRST, C28RSTIN	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3	WDT0 reset	Same as above	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	WDT1 reset	Same as above	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	MNMIWD	Same as above	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6	Master Software reset	SRXRST, C28RSTIN	Yes	Yes	Yes	Yes	No	Yes	Yes
7	Master Debugger Reset	SRXRST, C28RSTIN	Yes	Yes	Yes	Yes	No	Yes	Yes

Copyright © 2012–2019, Texas Instruments Incorporated

1.3.2.1.1 Reset Handling in Master System Boot ROM

ROM is mapped to address 0x0000 0000 on the master subsystem; this ROM will be referred to as M-Boot ROM in this document and all the supporting documents of this device. M-Boot ROM has an exception vector table with an initial stack pointer address, reset vector, and other exception/interrupt vectors located starting at 0x0000 0000, which is the default NVIC base address. Refer to the *Cortex-M3* manual for the NVIC exception table location and contents.

Whenever the master subsystem is reset, the Cortex M3 CPU fetches the reset vector address from the NVIC exception table and starts executing code in M-Boot ROM. This section gives an overview of how M-Boot ROM handles different reset causes.

Since the Cortex-M3 subsystem on this device is the master subsystem, it is brought out of reset first and will start executing code in M-Boot ROM while the analog and control subsystem are held under reset. M-Boot ROM software eventually brings both control and analog subsystems out of reset.

After a reset, the M-Boot ROM looks at the MRESC register to identify the reset cause. The following sections explain M-Boot ROM actions for each reset cause.

1.3.2.1.2 POR Reset Cause:

- M-Boot ROM initializes all master subsystem RAM(s) to 0x0000 0000. This will also zero-initialize RAM ECC and RAM PARITY bits for respective memories. Refer to the *Internal Memory* chapter for more details on RAM initialization.
- The PLL is left at its default state and M-Boot ROM modifies SYSDIVSEL to /1 and M3SSDIVSEL to /1 so PLLSYSCLK = M3SSCLK = OSCCLK.
  - Note that this means both the master subsystem and control subsystem are configured to run at the same frequency on power-up, so users have to be careful of this while selecting an OSCCLK frequency.
- Brings control and analog subsystems out of reset.
- Since a POR reset also causes an $\overline{\text{XRS}}$ reset, M-Boot ROM clears both the POR and $\overline{\text{XRS}}$ bits (bits 1 and 0 respectively) in the MRESC register and continues to boot. All the resets which come out on $\overline{\text{XRS}}$ as shown in Table 1-5 will set the $\overline{\text{XRS}}$ bit in the MRESC register. M-Boot ROM clears this bit if it was set, and continues execution as per the boot procedure detailed in the *Boot ROM* chapter.
  - The reason M-Boot ROM clears bit 0 and 1 in the MRESC register for this reset cause is because if there was a second reset for any other reason after the $\overline{\text{XRS}}$ reset, then boot ROM should not clear out the RAM contents or reset the clock dividers to default (for example, a debugger reset by the user while developing code on bench).

1.3.2.1.3 $\overline{\text{XRS}}$ Reset Cause

- M-Boot ROM will zero-initialize the stack memory needed for boot ROM execution, which is part of C2 RAM (0x2000 4004 - 0x2000 4900).
  - Note that location 0x2000 4000 on C2 RAM is not ZERO-initialized by M-Boot ROM except when a POR reset occurs because this location contains the boot status of the device for applications to read and handle different events that might occur during boot. More details on the boot status are given in the *Boot ROM* chapter.
- The PLL is left at its default state and M-Boot ROM modifies SYSDIVSEL to /1 and M3SSDIVSEL to /1 so PLLSYSCLK = M3SSCLK = OSCCLK.
  - Note that this means both the master subsystem and control subsystem are configured to run at the same frequency on power up, so users need to be careful of this while selecting an OSCCLK frequency.
- Brings control and analog subsystems out of reset.
- All the resets which come out on $\overline{\text{XRS}}$ as shown in Table 1-5 will set the $\overline{\text{XRS}}$ bit in the MRESC register. M-Boot ROM clears this bit if it was set, and continues execution as per the boot procedure detailed in the *Boot ROM* chapter.
  - The reason for M-Boot ROM to clear bit 0 in the MRESC register for this reset cause is that if there was a second reset for any other reason after the $\overline{\text{XRS}}$ reset, then boot ROM should not set clock dividers to default (for example, a debugger reset by the user while developing code on bench).

Copyright © 2012–2019, Texas Instruments Incorporated

1.3.2.1.4 Watchdog Timers Reset Cause (WDT0, WDT1, MNMIWD)

Watchdog Timer 0, Watchdog Timer 1, and the Master NMI WD timer resets will cause a low pulse on $\overline{\text{XRS}}$, which will reset the entire device. These resets are handled the same way as an $\overline{\text{XRS}}$ reset. Note that M-Boot ROM only clears the $\overline{\text{XRS}}$ bit (bit 0 in the MRESC register). It will not clear any other bits in the MRESC register to allow the user application to handle the actual reset cause if it occurred because of any WD timers.

1.3.2.1.5 Master Software Reset Cause

For a software reset cause, M-Boot ROM will zero initialize the stack memory for boot ROM execution and continues to boot. Clock settings are not changed by boot ROM for this reset cause.

1.3.2.1.6 Master Debugger Reset Cause

The master subsystem, the control subsystem, and shared resources are reset for this reset type. If M-BootROM is executed after this reset, the clock settings are not changed.

1.3.2.2 Control Subsystem Reset Handling

The C28x CPU is reset by the $\overline{\text{C28RSTIN}}$ signal and the C28x CPU, in turn, resets the rest of the control subsystem with the $\overline{\text{C28SYSRST}}$ signal.

The $\overline{\text{C28RSTIN}}$ signal has five possible sources: $\overline{\text{XRS}}$, C28NMIWD, M3SWRST, M3DBGRST, and $\overline{\text{M3RSNIN}}$. The C28NMIWD is set in response to time-out conditions of the C28x NMI Watchdog. The M3SWRST is a software-generated reset output by the NVIC. The M3DBGRS is a debugger-generated reset that is also output by the NVIC. The $\overline{\text{M3RSNIN}}$ reset comes from the Cortex-M3 subsystem to selectively reset the control subsystem from Cortex-M3 software. The C28x processor can learn the status of the internal $\overline{\text{ACIBRST}}$ reset signal and the external $\overline{\text{XRS}}$ pin by reading the DEVICECNF register.

As shown in Table 1-6, for the POR, $\overline{\text{XRS}}$, MWDT0, MWDT1, and MNMIWD resets, both the master and control subsystems are reset and the control subsystem is held in reset. Also, M-Boot ROM will bring the control subsystem out of reset and it will start executing code in its ROM when out of reset.

Master software and master debugger resets propagate to the control subsystem. The control subsystem will not be held in reset. Instead, it will restart executing software in the control subsystem boot ROM.

Note: For the control subsystem to see the master software and debugger resets, it should be in the RUN state. If the control subsystem is under DEBUG HALT and the master subsystem sends a debugger reset, the control subsystem will miss this reset.

The control subsystem core and control subsystem peripherals are only reset by the C28 (control subsystem) debugger and NMIWD resets. When the control subsystem is reset by CNMIWD reset, it will reset and restart running software code in the control subsystem ROM. In addition, an NMI is generated to the master subsystem indicating that the control subsystem is reset by an NMI. If the master subsystem does not handle or acknowledge this NMI, the master subsystem is also reset, and a bit is set in the MRESC register which tells which C28NMI was unserviced that caused a reset.

Unlike the master subsystem which has a reset cause register, the control subsystem does not.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-6. Control Subsystem Resets, Signals and Effects

No.	Reset Source	C28x Core Reset	JTAG Reset	Control Subsystem Reset	Control Subsystem Held in Reset
1	POR	Yes	Yes	Yes	Yes
2	$\overline{\text{XRS}}$	Yes	Yes	Yes	Yes
3	MWDT0 reset	Yes	Yes	Yes	Yes
4	MWDT1 reset	Yes	Yes	Yes	Yes
5	MNMIWD	Yes	Yes	Yes	Yes
6	Master Software reset	Yes	Yes	Yes	No
7	Master Debugger Reset	Yes	Yes	Yes	No
8	C28 Debugger Reset ($\overline{\text{SYSRS}}$)	Yes	No	Yes	No
9	C28 NMIWD Reset	Yes	No	Yes	No

1.3.2.2.1 Reset Handling in Control Subsystem Boot ROM

The C28x CPU core of the control subsystem starts executing code from the reset vector located at 0x3FFFC0. ROM is mapped to this location of the memory map, and this location points to the reset vector located in ROM. The ROM associated with the control subsystem will be referred to as C-Boot ROM in this document.

Whenever the control subsystem is reset because of any of the reset sources as mentioned in Table 1-6, the CPU will start executing code from C-Boot ROM. C-Boot ROM uses the first 0x180 locations of M0 RAM for its execution stack so on start up, C-Boot ROM will ZERO-INIT the entire M0 RAM. This is done to clear unwanted RAM ECC errors which might pop-up by default on a reset or power-up. It is up to the application to reset other RAM(s) of the control subsystem. C-Boot ROM then enables the PIE and installs interrupt handlers for handling IPC communication from the master subsystem and enters IDLE mode. C-Boot ROM wakes up if there is an IPC interrupt from the master subsystem, handles the IPC communication, and goes back to IDLE mode. Please refer to the *Boot ROM* chapter for more details on the C-Boot ROM procedure and supported IPC communication.

If there is an NMI during the execution time of C-Boot ROM (except for a Missing Clock NMI), C-Boot ROM acknowledges the NMI, sends an IPC to the master subsystem, updates the status in CTOMBOOTSTS register, and enters a while(1) loop waiting for the master subsystem to handle the NMI error condition. A missing clock NMI is triggered to both the master and control subsystems, but in this event, the control subsystem will acknowledge this NMI and return back to what it was doing before the interrupt.

Refer to Section 1.5.6.1 for more details on CNMI sources and how C-Boot ROM handles each NMI.

In short, the control subsystem acts as a slave to the master subsystem by going to IDLE mode and waiting for the master subsystem's command on how to proceed further.

1.4 WIR Mode

This device supports a wait-in-reset mode which is primarily used to hold the master and control subsystem CPU cores in a known active state prior to initiating a Flash programming procedure. This is especially important on "fresh" devices where the Flash is not programmed. This mode is also important in avoiding accidently tripping security mechanisms before the debugger has had a chance to connect to the CPUs.

Both the master and control subsystems each have a WIR mode register; MWIR for the master subsystem WIR mode register and CWIR for the control subsystem WIR mode register. These registers are read by M-Boot ROM and C-Boot ROM respectively, each time boot ROM is run after any kind of reset.

Both MWIR and CWIR registers are latched in with the state of the same EMU0 and EMU1 pins on $\overline{\text{XRS}}$, or whenever the SAMPLE bit (bit 2) is set in the respective WIR mode register. See the MWIR and CWIR register descriptions in the *System Control Registers* section for more details.

Table 1-7 shows which of the different possible values on EMU0 and EMU1 pins can put the device in WIR mode.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-7. EMU0/1 Pin Values for WIR Mode

EMU0	EMU1	WIR mode
0	0	NO
0	1	WIR_MODE_YES
1	0	NO
1	1	NO

1.4.1 Entering WIR Mode

In the boot procedure, M-Boot ROM (after releasing the control subsystem and ACIB out of reset), reads EMU0 (bit 0) and EMU1 (bit 1) bits of the MWIR register. If the values match the WIR_MODE_YES value as per Table 1-7 then M-Boot ROM enters a wait forever loop as shown in Figure 1-2.

C-Boot ROM is executed after the control subsystem is out of reset. C-Boot ROM reads EMU0 and EMU1 bits of the CWIR register. If the values match the WIR_MODE_YES values as per the Table 1-7, then C-Boot ROM enters a wait forever loop as shown in Figure 1-2.

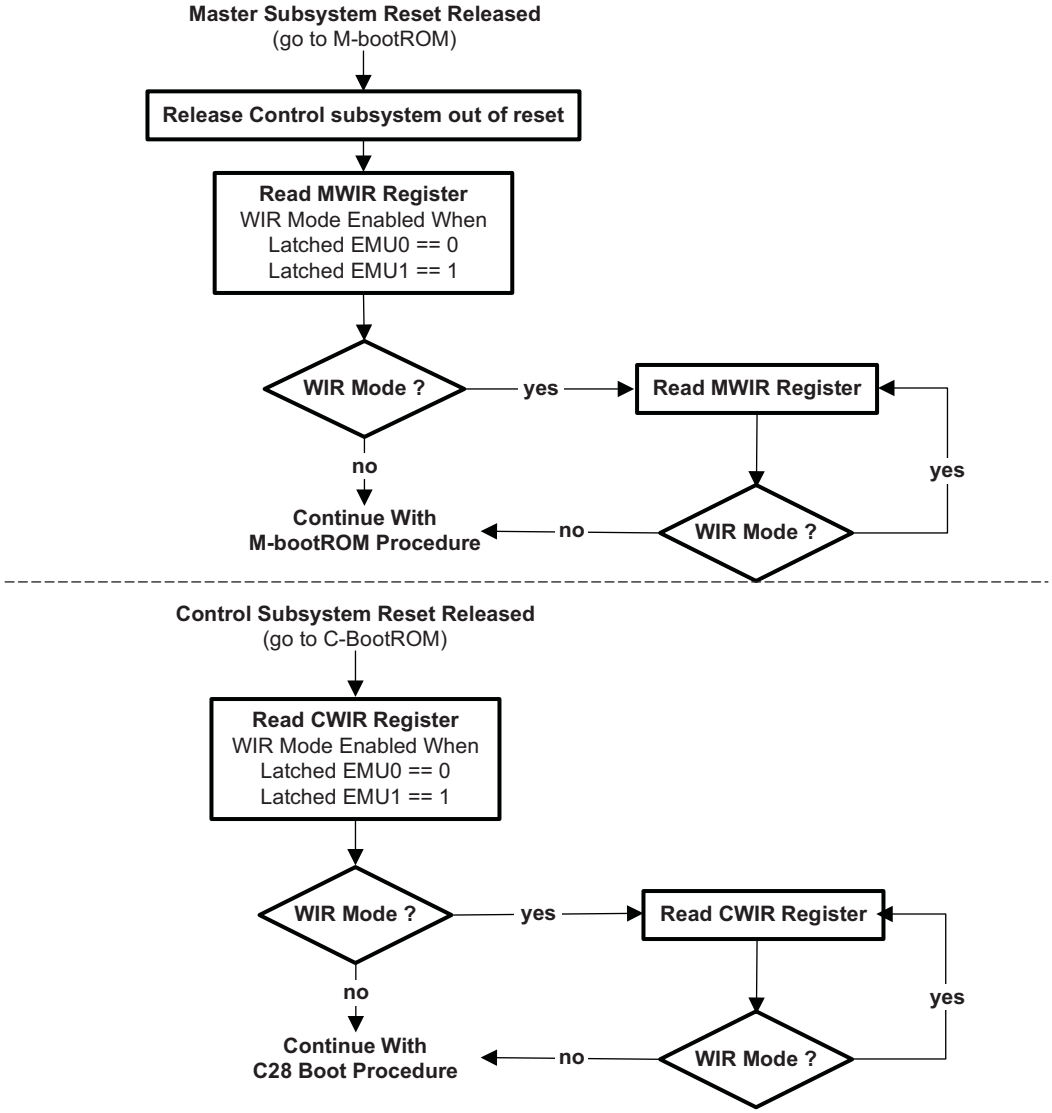
There is more than one way to put the master subsystem and/or control subsystem in WIR mode. The user has to make sure that EMU0 and EMU1 bits in the MWIR register and/or the CWIR register are set to the WIR_MODE_YES value and run the boot ROM. To achieve this, the user can:

- Set EMU0 and EMU1 pins on the device to the WIR_MODE_YES value and give an $\overline{\text{XRS}}$ reset to the device. This will latch the state of EMU0 and EMU1 pins in the WIR registers and execute the boot ROM(s) in each subsystem. Please see the device-specific data manual for more details on the pin locations.
- Directly write the WIR_MODE_YES value to the EMU0 and EMU1 bits in the MWIR and CWIR registers. This will put both the master and control subsystems in WIR mode when boot ROM(s) are run after reset. Both EMU0 and EMU1 are R/W type bits and are reset to the actual state of EMU0 and EMU1 pins on $\overline{\text{XRS}}$ reset or whenever the sample bit is set in the respective WIR registers to re-sample these pins. So, it is not necessary to set the sample bit and force a software reset or debugger reset.
- Set EMU0 and EMU1 pins to the WIR_MODE_YES values and set the sample bit in each WIR mode register and give a software reset or debugger reset to the device.

Figure 1-2 shows how each boot ROM puts respective CPU in WIR mode.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-2. Master and Control Subsystem WIR Mode Flow



1.4.2 Exiting WIR Mode

As explained earlier, after the master subsystem and/or control subsystem enter WIR mode, they stay in this mode as long as the EMU0 and EMU1 bits in the respective WIR register are set to match the WIR_MODE_YES value. If they do not match the WIR_MODE_YES value, then the boot ROM continues its execution by exiting the WIR mode. Although both the boot ROMs in the master subsystem and control subsystem continue to read the WIR mode register, they do not force the EMU0 and EMU1 bits to re-latch the EMU0 and EMU1 pin status by setting the sample bits. So in order to force re-latching of the EMU0 and EMU1 bits of WIR mode registers, the user has to give an $\overline{\text{XRS}}$ reset or has to set the re-sample bit from the debugger.

Below are different options for bringing the device out of WIR mode after it enters the mode.

- Set EMU0 and EMU1 pins on the device to a value other than the WIR_MODE_YES value as shown in Table 1-7 and give an $\overline{\text{XRS}}$ reset to the device. Refer to the device-specific data manual for more details on the pin locations.
- EMU0 and EMU1 bits in MWIR and CWIR registers are R/W type bits and are reset to the actual state of EMU0 and EMU1 pins on $\overline{\text{XRS}}$ reset or whenever the sample bit is set in the respective WIR registers to re-sample these pins. The user can then directly write a value other than the WIR_MODE_YES value (refer to Table 1-7) to these bits, and not set the sample bit. Therefore, when

Copyright © 2012–2019, Texas Instruments Incorporated

each boot ROM reads their respective WIR mode register it will exit WIR mode and continue to boot or give a software or debugger reset to the subsystem.

- Set EMU0 and EMU1 pins to a value other than the WIR_MODE_YES value and set the sample bit in each WIR mode register. The next time boot ROM reads this register, it will exit the WIR mode and continue to boot or give a software reset or debugger reset to the device and restart boot ROM execution.

1.5 Exceptions and Interrupts Control

This device has exception capturing and handling ability which enables the device to be used in many safety critical applications. Device run-time exceptions that can be detected and acted upon include clock failure detection, memory access errors detection, bus fault detection, and interrupt handler address mismatch errors. This device also supports back-to-back non-maskable interrupt handling capability along with highly configurable peripheral interrupt handling.

The master subsystem is built around the Cortex-M3 ARM core and has the Nested Vectored Interrupt Controller (NVIC) module, while the control subsystem is built around the C28x core and has the peripheral interrupt expansion (PIE) module. This enables the user to configure, handle, and serve interrupt requests from different subsystem peripherals and handle various exceptions that can occur in the device during its operation. The master subsystem is capable of handling exceptions on the control subsystem and ACIB, whereas the control subsystem is capable of handling exceptions related to control peripherals. Exception handling on the master subsystem is built such that, it will be able to identify and handle the errors even if the control subsystem fails to handle its exceptions.

This section provides details about interrupts and exception handling supported by both the master and control subsystems. Both the master and control subsystems each have an independent NMI module which captures various exceptions that can occur in the system and trigger an NMI to the respective CPU core.

1.5.1 Master Subsystem Nested Vectored Interrupt Controller

Refer to the NVIC section of *the Cortex-M3 Peripherals* chapter of this document for more details on exceptions and interrupts supported by the NVIC in the master subsystem.

As part of the master subsystem, the Cortex-M3 NVIC handles exceptions that can occur on the control subsystem and ACIB. The Cortex-M3 NVIC module supports a non-maskable interrupt, which has higher priority than all other NVIC supported interrupts or exceptions. The master subsystem's non-maskable interrupt module (MNMI) is responsible for generating this non-maskable interrupt to the Cortex-M3 CPU core in the master subsystem. Refer to Section 1.5.3 and Section 1.5.6 for more details on NMI handling.

The NVIC supports a HARDFAULT exception interrupt which has higher priority than any programmable interrupts but less priority than an NMI. Other programmable exceptions supported by the NVIC are memory management faults, bus faults, and usage fault programmable exceptions. These programmable exceptions are disabled by default and the system errors which can cause these exception events end up triggering a HARDFAULT exception. Section 1.5.2 provides details on the events that cause these exceptions.

On power-up and any reset that resets the master CPU, the NVIC is mapped to the address of 0x0000 0000 in ROM. The M-Boot ROM installs pre-defined interrupt handlers in the default NVIC table as needed for the boot ROM execution and control. Once the user application is started by boot ROM, they should have their own NVIC vector table and map the NVIC base address to user locations. If users fail to re-map the NVIC to their application needs, any interrupt that occurs while the application is executing ends up calling interrupt and exception handlers installed by boot ROM. Refer to the *Boot ROM* chapter for more details on boot ROM handlers in the NVIC.

NOTE: NVIC Interrupt No. 89 (System/USB PLL OutOfLock) is generated whenever the system PLL or USB PLL is no longer locked. The interrupt handler should check the status bits to find out which PLL is no longer locked, and the master subsystem should indicate out-of-lock status to the control subsystem using an IPC. This has to be taken care of by the user as per the application requirements.

Copyright © 2012–2019, Texas Instruments Incorporated

1.5.2 Master Subsystem Exceptions Handling

Table 1-8 provides information on different exceptions supported by NVIC on the master subsystem.

Table 1-8. Master Subsystem Exceptions

Exception Type	Vector Number	Priority	Description
Reset	1	-3 (highest)	This exception is invoked on power up and on any other reset. On the first instruction, Reset drops to the lowest priority and then is called the base level of activation. This exception is asynchronous.
Non-Maskable Interrupt(NMI)	2	-2	This exception is caused by the assertion of the NMI signal or by using the NVIC Interrupt Control State register and cannot be stopped or pre-empted by any exception but Reset. This exception is asynchronous. Used for clock fail condition, and external M3 side GPIO NMI input.
Hard Fault	3	-1	This exception is caused by all classes of Fault when the fault cannot activate due to priority or the configurable fault handler has been disabled. This exception is synchronous.
Memory Management	4	Programmable	This exception is caused by an MPU mismatch, including access violation and no match. This exception is synchronous.
Bus Fault	5	Programmable	This exception is caused by a pre-fetch fault, memory access fault, or other address/memory related faults. This exception is synchronous when precise and asynchronous when imprecise. This fault can be enabled or disabled. Generated for memory access errors and RAM and flash uncorrectable data errors in the device.
Usage Fault	6	Programmable	This exception is caused by a usage fault, such as an undefined instruction executed or an illegal state transition attempt. This exception is synchronous.

From the above exceptions, the NMI and bus faults are generated by the digital subsystem, whereas memory management errors are generated internally by the M3 MPU.

Bus Fault Exceptions:

For all uncorrectable memory errors during M3 CPU reads or writes (address, parity or double data error), HRESP-based, and HREADY-based error responses will be generated by the memory control logic.

Write Accesses:

When an HRESP-error is generated for write accesses, if the intended write was a stack push, STKERR status will get set by the NVIC upon seeing the bus fault indication for the stack push operation. If the application so prefers, it can treat a STKERR as a low priority exception and pend the same, except when stacking for an exception.

Read Accesses:

For all uncorrectable errors during reads, the same HRESP-error indication will be generated. The M3 core will use this error indication in the following manner:

- For bus errors during normal data reads, the M3 would bus fault, but the application can treat this as a lower priority, thereby enabling a higher priority exception to be serviced by the CPU. As an example, there may be a bus fault in a user thread and still the CPU could service an interrupt to handle critical interrupts - the bus fault can be handled afterwards. But if the read occurs during an ISR bus fault, it will be treated as a hard fault, since it is in code for an exception that should never fault.
- For bus errors during stack pops, the NVIC will set the UNSTKERR status indication and the M3 will bus fault
- For bus errors during vector table reads, the NVIC will set VECTTBL and the M3 will bus fault. If there is a double fault when the bus fault handler tries to read the vector, the M3 CPU will enter a LOCKUP condition. In this condition, the M3 WDT timers will time out and issue a reset to the system.
- For bus errors during fetches, even if the M3 sees an error response, it will not internally bus fault

Copyright © 2012–2019, Texas Instruments Incorporated

unless the CPU is decoding the erroneous instruction fetched

For more details on how the remaining exceptions are generated by the Cortex-M3 CPU, refer to the NVIC section of the *Cortex-M3 Peripherals* chapter.

Refer to the *Boot ROM* chapter for more details on how boot ROM handles HARDFAULT exceptions if it occurs during boot ROM execution.

On the master subsystem all the below errors generate a BUSFAULT.

- RAMUNCERR - RAM Uncorrectable error. This is a double bit error generated by the RAM wrapper logic as a bus error. Refer to *the Internal Memory* chapter of this document for more details.
- RAMACCVIOL - Ram Access violation. Refer to the *Internal Memory* chapter for more details on what can generate this error.
- FLASHUNCERR - Flash Uncorrectable error. Refer to the *Internal Memory* chapter for more details on this error.

1.5.3 Master Subsystem Non-Maskable Interrupt (MNMI) Module

The master subsystem has the capability of detecting all serious errors that could occur in the entire system including all the subsystems, and inform the main CPU core about the error. An NMI exception to the M3 CPU on the master subsystem will be generated only when at least one or more of the below NMI error sources become active. More details on each of the sources is given in Section 1.5.3.1.

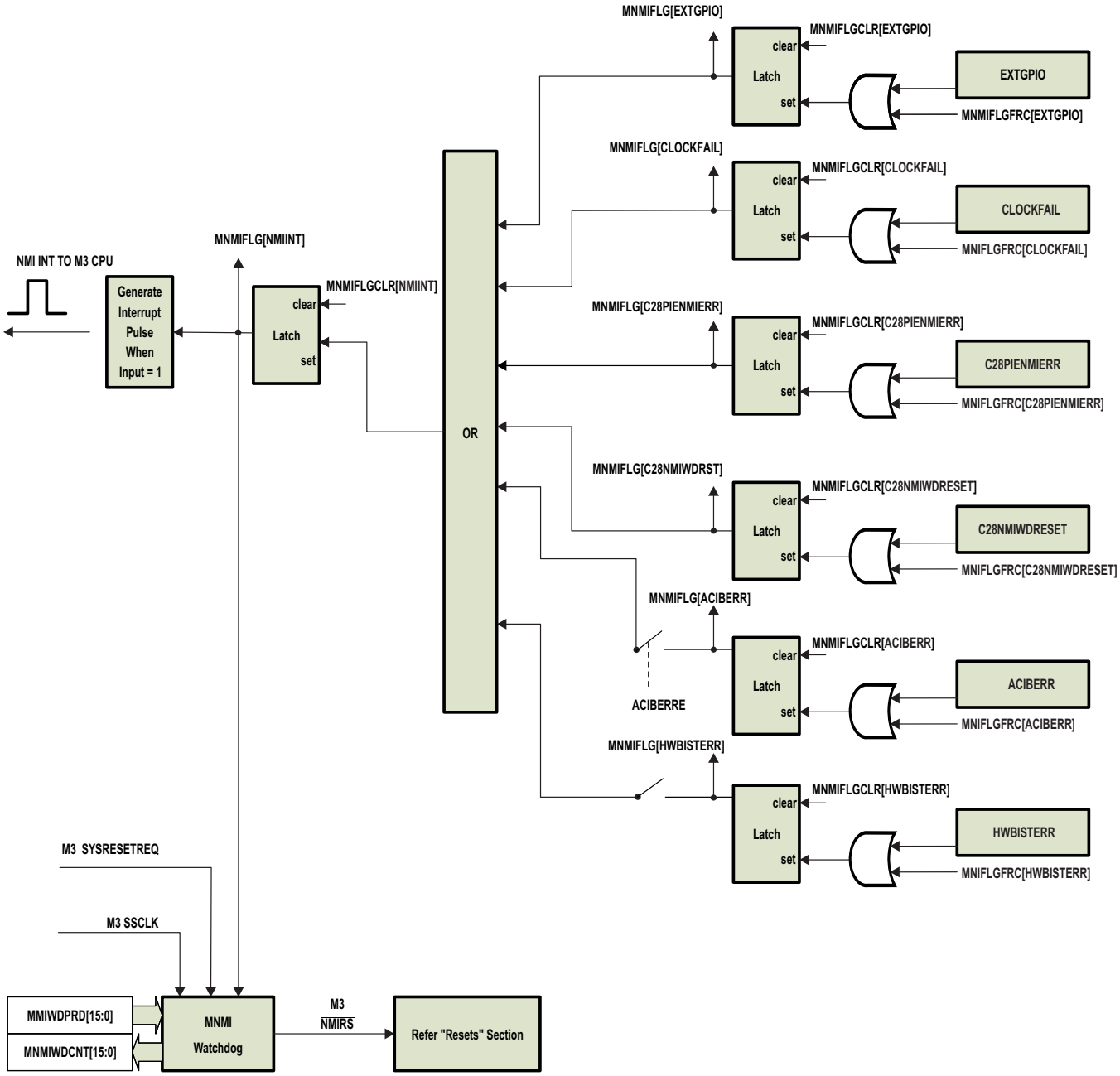
1. Clock fail condition detected
2. External GPIO input signal is requesting an NMI
3. Error condition is generated on a C28 PIE NMI vector fetch
4. CNMIWD timed out and issued a reset to the C28 CPU
5. Stuck condition is detected on the ACIB INTS or READY signals (referred to as ACIBERR in this section of this spec)
6. Voltage Regulator warning NMI

All these NMI sources are "OR-ed" to generate the NMI input to the M3 NVIC. The NMI triggers a MNMIWD counter running at the master subsystem frequency. The MNMIWD counter will stop counting only if all the pending NMIs are acknowledged by clearing the pending flags in the MNMIFLG register. If the pending NMI is not acknowledged before the MNMIWD counter reaches the value programmed in the NMI WatchDog period register (MNMIWDPRD), an NMIWD reset is generated to the master subsystem, which will reset the entire device.

Figure 1-3 shows different sources that can trigger an NMI to the Cortex-M3 on the master subsystem and the registers associated with them.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-3. Master Subsystem NMI Sources and MNMIWD



All the NMI sources except for the ACIBERR NMI shown in Figure 1-3 are enabled by default on reset. There is no provision for the user to disable NMI sources that are enabled by default. The ACIBERR NMI can be enabled by setting the ACIBERRE (bit 9) bit in the MNMICFG register. Once this bit is set in the MNMICFG register, it cannot be cleared by the user, and only a master subsystem reset can clear it. See the MNMICFG register for more details.

Whenever an NMI signal is generated, the respective bit in the MNMIFLG register is set. To aid in debug, development, and testing, an MNMIFLGFRC register is provided. Setting these bits in this register will force the NMI to the CPU core as shown in Figure 1-3. Refer to the MNMIFLGFRC register for more details. When an NMI is triggered to the master CPU, an MNMIWD counter is triggered and begins counting. It will reset the device when the MNMIWD counter reaches a programmed MNMIWD period value. The MNMIWD counter will stop counting and reset back to zero once all the set MNMIFLG bits and the NMIINT flag bit in the MNMIFLG register are cleared.

Copyright © 2012–2019, Texas Instruments Incorporated

1.5.3.1 Master Subsystem NMI Sources

This section explains all the possible NMI sources on the master subsystem.

1.5.3.1.1 Clock Fail Condition

A main oscillator verification circuit is provided that generates an error condition if the oscillator is running too fast or too slow or goes missing. This logic is referred to as Missing Clock Detection logic and is explained in detail in Section 1.6.2. When a missing clock error is generated, the CLOCKFAIL bit (bit 1) of the MNMIFLG register is set, the clock source is switched to the 10 MHz internal oscillator, and the PLL is bypassed.

The CLOCKFAIL NMI is triggered to both the master and control subsystems. Since this NMI is enabled by default on power up, it makes it necessary for boot ROM to handle this NMI. Refer to the *Boot ROM* chapter of this document for more details on how boot ROM handles this NMI.

1.5.3.1.2 External GPIO Input Signal is Requesting an NMI

The NMI signal is the alternate function for GPIO port pin, PB7. The alternate function must be enabled in the GPIO register for the signal to be used as an interrupt, as described in the *GPIOs* chapter of this document. Note that enabling the NMI alternate function requires the use of the GPIO lock and commit function. The NMI signal is active high and asserts the enabled NMI signal above VIH to initiate the NMI interrupt sequence.

Since this NMI is enabled by default on power-up, it is necessary for boot ROM to handle this NMI. The M-Boot ROM does not configure GPIO PB7 for this NMI function on power-up, so it is not possible for this NMI to occur while boot ROM is executing after a POR or $\overline{\text{XRS}}$ reset. However, the user application can configure PB7 for NMI operation, give a warm reset, and execute through boot ROM. In which case this NMI could occur while M-Boot ROM is executing. However, the boot ROM simply ignores this NMI by clearing the required NMI flags and continues the boot process. Refer to the *Boot ROM* chapter for more details.

1.5.3.1.3 Error Condition Generated on PIE NMI Vector Fetch in the Control Subsystem

Please refer to Section 1.6.4 for more details on how this NMI can be generated.

This NMI is enabled by default on power up and the user cannot disable this NMI. Please refer to the *Boot ROM* chapter for more details on how boot ROM handles this NMI.

1.5.3.1.4 CNMIWD Timed Out and Issued a Reset to C28 CPU

The Cortex-M3 CPU on the master subsystem can detect if the control subsystem is reset by CNMIWD using this NMI. The control subsystem is not held in reset when CNMIWD causes a reset, and will reset and continue executing C-Boot ROM. However, the master subsystem can choose to reset the control subsystem and hold the control subsystem in reset using the CRESCNF register. The master subsystem will only know that the control subsystem was reset because it was not able to handle an NMI. It will not know specific details on which NMI was unserviced and why, but considering any NMI on the control subsystem is serious, effective action can be taken by the user on the master subsystem.

This NMI is enabled by default on the master subsystem. Refer to the *Boot ROM* chapter on how boot ROM handles this NMI if it occurrs during the execution time of boot ROM.

1.5.3.1.5 ACIBERR NMI

This NMI is disabled by default and the user can enable this NMI by setting the ACIBERRE bit in the MNMICFG register. However, even when this NMI is disabled (ACIBERRE bit in MNMICFG register is "0"), if the error condition that triggers this NMI occurs, the ACIBERR bit is set in the MNMIFLG register. This error is generated if a stuck condition is detected on the ACIB INTS or READY signals.

Copyright © 2012–2019, Texas Instruments Incorporated

First, whenever a READ or WRITE strobe is generated, the READY signal is sampled on the very first cycle. The READY signal should be low. If it is not low, then the ACIBERR NMI is triggered to both the master and control subsystems irrespective of which subsystem issued the current READ or WRITE. Second, if a read or write access is generated and the active cycle is not completed by a specified timeout, then this indicates either READY was not generated or that READY is stuck high. The timeout starts whenever a READ or WRITE access is activated across the ACIB. This time-out check is internally implemented in ACIB hardware.

The INTS strobe signal is pulsed high for only one cycle. If this is not the case, this is an error and a pulse is generated on ACIBERRNMI that is fed to both the master and control subsystems' NMI modules.

Apart from READ/WRITE and interrupts that pass through the ACIB, there are some trigger signals that could get stuck high. In this case, triggers will get continually exported on the ACIB. If this happens, ACIB accesses will get stalled which will then eventually cause the READY time-out to occur. Hence, the READY timeout mechanism will capture this case.

The status of the READY and INTS signals' state can be read by both the master subsystem and control subsystem by reading the MCIBSTATUS and CCIBSTATUS read-only registers.

An NMI is generated to both the master subsystem and control subsystem on this error condition.

1.5.3.2 Master Subsystem NMIWD Module

As explained previously, the master subsystem is equipped with an NMI Watchdog module whose function is to make sure that a triggered non-maskable interrupt is handled by user software. This can be achieved by clearing the error conditions and clearing the respective flags in the MNMIFLG register or by acknowledging the NMI and gracefully shutting down the system. If none of the actions mentioned are taken, then the MNMIWD counter keeps counting until the counter value reaches the MNMIWD period register value. An MNMIWD reset will then be generated, which will reset the entire device. Please refer to Section 1.3.1.5 for more details on the reset.

As shown in Figure 1-3 above, any enabled NMI source can set the NMIINT respective bit in MNMIFLG register, which will trigger an NMI to the CPU and start the NMIWD counter. The NMIWD counter will keep counting as long as the NMIINT bit of the MNMIFLG register is not cleared or a reset is generated.

The MNMIWD counter is clocked by the M3 system clock. The MNMIWDPRD register, which is the MNMI Watchdog Period register, can be programmed with a period limit as per user requirements which sets the clock cycle limit required for software to handle or acknowledge the NMI. A timeout condition that generate the NMI watchdog reset means that the counter value of the MNMIWDCNT register reached the value programmed in the period register, MNMIWDPRD.

1.5.3.2.1 Emulation Considerations

When the Cortex-M3 CPU is suspended (in debug halt), the NMI watchdog counter will be suspended.

1.5.3.3 Handling of MNMI

User software must clear all the flag bits which are set in the MNMIFLG register before clearing NMIINT, bit 0 of the MNMIFLG register. If the user clears the NMIINT bit in the MNMIFLG register before clearing all the individual flag bits, as soon as the NMIINT bit is cleared it will be set back to "1" again. This will generate another back-to-back NMI to the master subsytem's CPU, and the NMIWD counter will start counting again.

1.5.4 Control Subsystem PIE

The control subsystem supports the peripheral interrupt expansion block to handle different exceptions and interrupts that occur on control subsystem modules during the device operation. The PIE enables multiplexing of numerous interrupt sources into a smaller set of interrupt inputs. The PIE block can support 96 individual interrupts that are grouped into blocks of eight. Each group is fed into one of 12 core interrupt lines (INT1 to INT12). Each of the 96 interrupts is supported by its own vector stored in a

Copyright © 2012–2019, Texas Instruments Incorporated

dedicated RAM block that is modifiable. The CPU, upon servicing the interrupt, automatically fetches the appropriate interrupt vector. It takes nine CPU clock cycles to fetch the vector and save critical CPU registers. Therefore, the CPU can respond quickly to interrupt events. Prioritization of interrupts is controlled in hardware and software. Each individual interrupt can be enabled/disabled within the PIE block.

By default on power-up or on reset (any reset which can reset the control subsystem CPU), the PIE is disabled and none of the peripheral interrupts are enabled. However, an NMI and ITRAP exceptions are enabled by default on power up and a reset occur. When the PIE is disabled, any NMI and ITRAP exception will use the handlers registered in the ROM vector table, and when the PIE is enabled, an NMI and ITRAP exceptions will use the handler's registers in the PIE vector table, which is located in PIE RAM. More details are explained in the following sections.

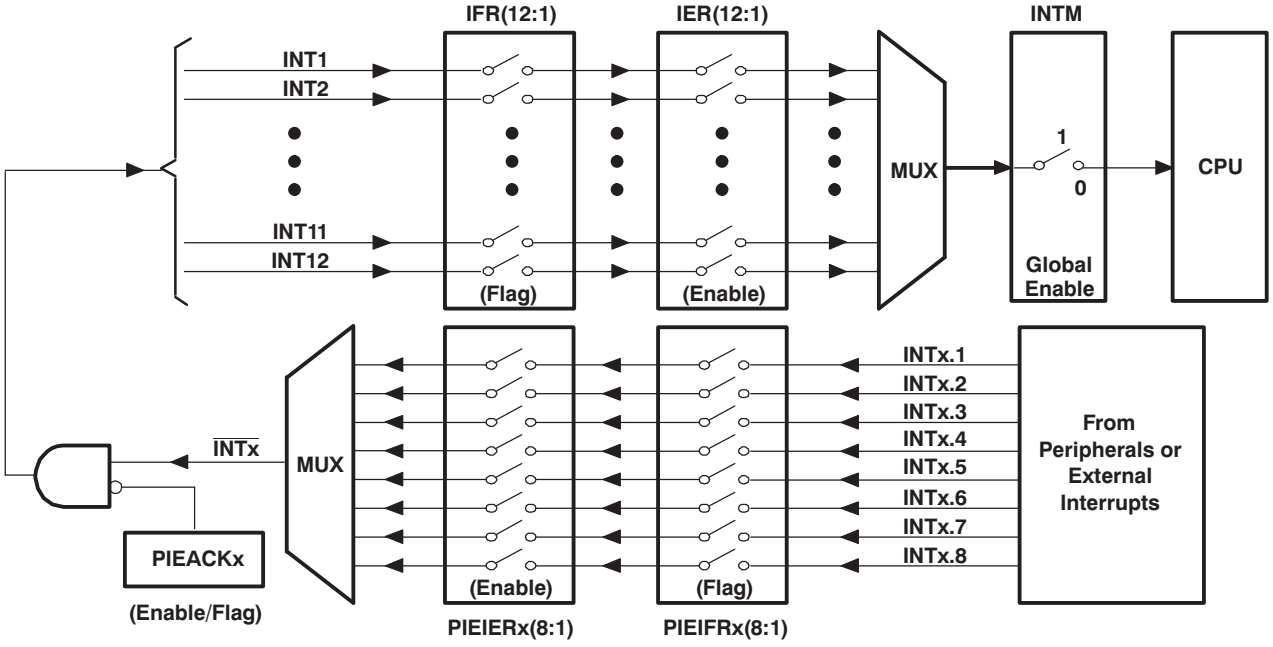
1.5.4.1 Overview of the PIE Controller

The 28x CPU supports one nonmaskable interrupt (NMI) and 16 maskable prioritized interrupt requests (INT1-INT14, RTOSINT, and DLOGINT) at the CPU level. The 28x devices have many peripherals and each peripheral is capable of generating one or more interrupts in response to many events at the peripheral level. Because the CPU does not have sufficient capacity to handle all peripheral interrupt requests at the CPU level, a centralized peripheral interrupt expansion (PIE) controller is required to arbitrate the interrupt requests from various sources such as peripherals and other external pins. The PIE vector table is used to store the address (vector) of each interrupt service routine (ISR) within the system. There is one vector per interrupt source including all MUXed and nonMUXed interrupts. The user populates the vector table during device initialization and updates it during operation.

1.5.4.1.1 Interrupt Operation Sequence

Figure 1-4 shows an overview of the interrupt operation sequence for all multiplexed PIE interrupts. Interrupt sources that are not multiplexed are fed directly to the CPU.

Figure 1-4. PIE Interrupts Multiplexing



Copyright © 2012–2019, Texas Instruments Incorporated

1.5.4.1.2 Peripheral Level

If an interrupt-generating event occurs in a peripheral, the interrupt flag (IF) bit corresponding to that event is set in a register for that particular peripheral.

If the corresponding interrupt enable (IE) bit is set, the peripheral generates an interrupt request to the PIE controller. If the interrupt is not enabled at the peripheral level, then the IF remains set until cleared by software. If the interrupt is enabled at a later time, and the interrupt flag is still set, the interrupt request is asserted to the PIE. Interrupt flags within the peripheral registers must be manually cleared.

1.5.4.1.3 PIE Level

The PIE block multiplexes eight peripheral and external pin interrupts into one CPU interrupt. These interrupts are divided into 12 groups: PIE group 1 - PIE group 12. The interrupts within a group are multiplexed into one CPU interrupt. For example, PIE group 1 is multiplexed into CPU interrupt 1 (INT1) while PIE group 12 is multiplexed into CPU interrupt 12 (INT12). Interrupt sources connected to the remaining CPU interrupts are not multiplexed. For the non-multiplexed interrupts, the PIE passes the request directly to the CPU.

For multiplexed interrupt sources, each interrupt group in the PIE block has an associated flag register (PIEIFRx) and enable (PIEIERx) register (x = PIE group 1 - PIE group 12). Each bit, referred to as y, corresponds to one of the eight MUXed interrupts within the group. Thus PIEIFRx.y and PIEIERx.y correspond to interrupt y (y = 1-8) in PIE group x (x = 1-12). In addition, there is one acknowledge bit (PIEACK) for every PIE interrupt group referred to as PIEACKx (x = 1-12).

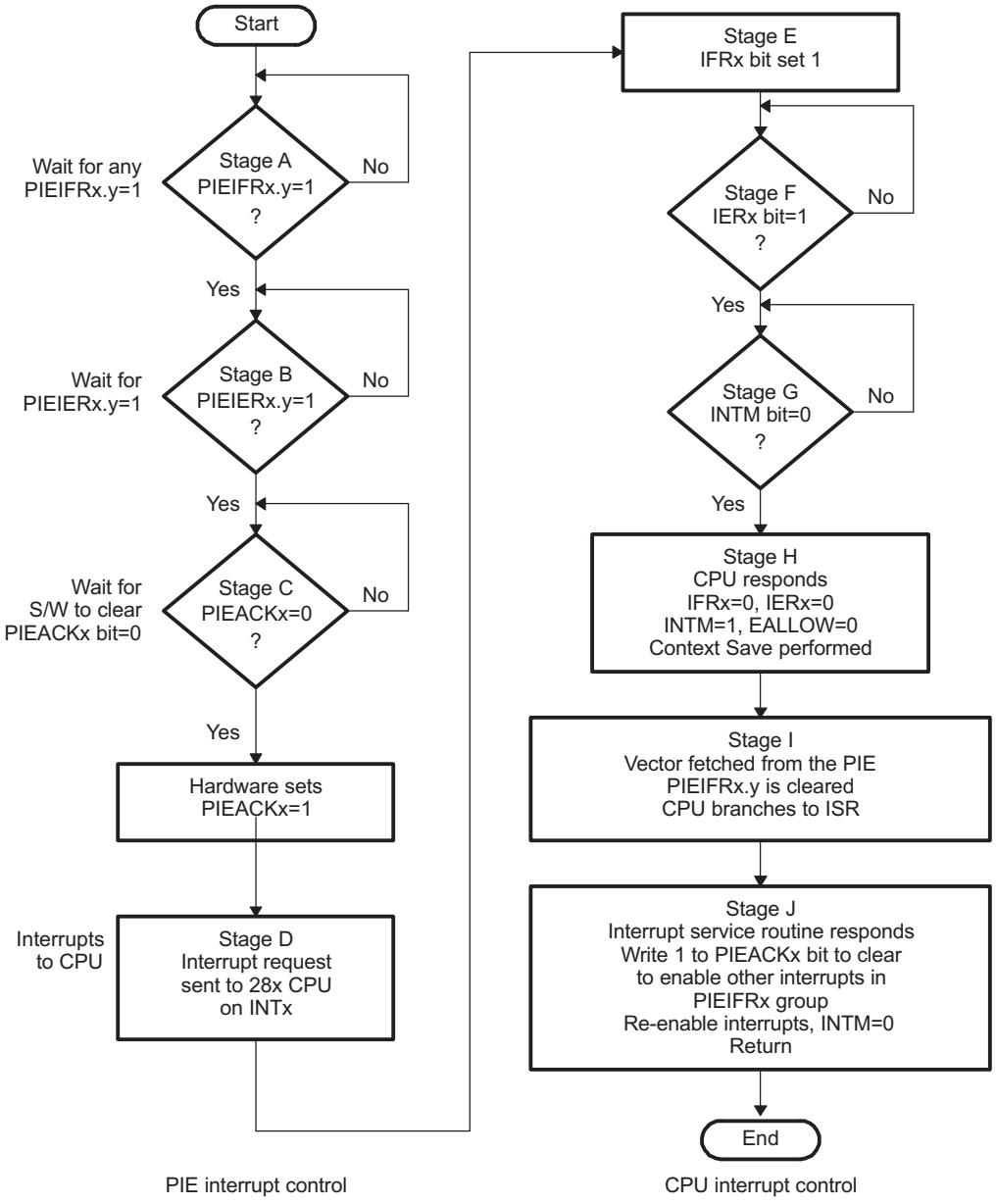
Once the request is made to the PIE controller, the corresponding PIE interrupt flag (PIEIFRx.y) bit is set. If the PIE interrupt enable (PIEIERx.y) bit is also set for the given interrupt then the PIE controller checks the corresponding PIEACKx bit to determine if the CPU is ready for an interrupt from that group. If the PIEACKx bit is clear for that group, then the PIE controller sends the interrupt request to the CPU. If PIEACKx is set, then the PIE waits until it is cleared to send the request for INTx.

1.5.4.1.4 CPU Level

Once the request is sent to the CPU, the CPU level interrupt flag (IFR) bit corresponding to INTx is set. After a flag has been latched in the IFR, the corresponding interrupt is not serviced until it is appropriately enabled in the CPU interrupt enable (IER) register or the debug interrupt enable register (DBGIER) and the global interrupt mask (INTM) bit.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-5. CPU Level Interrupt Handling



NOTE: For multiplexed interrupts, the PIE responds with the highest priority interrupt that is both flagged and enabled. If there is no interrupt both flagged and enabled, then the highest priority interrupt within the group (INTx.1 where x is the PIE group) is used.

As shown in Table 1-9, the requirements for enabling the maskable interrupt at the CPU level depends on the interrupt handling process being used. In the standard process, which happens most of the time, the DBGIER register is not used. When the 28x is in real-time emulation mode and the CPU is halted, a different process is used. In this special case, the DBGIER is used and the INTM bit is ignored. If the DSP is in real-time mode and the CPU is running, the standard interrupt-handling process applies.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-9. Enabling Interrupt

Interrupt Handling Process	Interrupt Enabled If"¦
Standard	INTM = 0 and bit in IER is 1
DSP in real-time mode and halted	Bit in IER is 1 and DBGIER is 1

The CPU then prepares to service the interrupt. This preparation process is described in detail in *TMS320x28x DSP CPU and Instruction Set Reference Guide* (literature number SPRU430). In preparation, the corresponding CPU IFR and IER bits are cleared, EALLOW and LOOP are cleared, INTM and DBGM are set, the pipeline is flushed and the return address is stored, and the automatic context save is performed. The vector of the ISR is then fetched from the PIE module. If the interrupt request comes from a multiplexed interrupt, the PIE module uses the PIEIERx and PIEIFRx registers to decode which interrupt needs to be serviced.

The address for the interrupt service routine that is executed is fetched directly from the PIE interrupt vector table. There is one 32-bit vector for each of the possible 96 interrupts within the PIE. Interrupt flags within the PIE module (PIEIFRx.y) are automatically cleared when the interrupt vector is fetched. The PIE acknowledge bit for a given interrupt group, however, must be cleared manually when ready to receive more interrupts from the PIE group.

1.5.4.2 Vector Table Mapping

On the control subsystem, the interrupt vector table can be mapped to four distinct locations in memory. In practice only the PIE vector table mapping is used.

This vector mapping is controlled by the following mode bits/signals:

VMAP	VMAP is found in Status Register 1 ST1 (bit 3). A device reset sets this bit to 1. The state of this bit can be modified by writing to ST1 or by SETC/CLRC VMAP instructions. For normal operation leave this bit set.
M0M1MAP	M0M1MAP is found in Status Register 1 ST1 (bit 11). A device reset sets this bit to 1. The state of this bit can be modified by writing to ST1 or by SETC/CLRC M0M1MAP instructions. For normal device operation, this bit should remain set. M0M1MAP = 0 is reserved for TI testing only.
ENPIE	ENPIE is found in PIECTRL Register (bit 0). The default value of this bit, on reset, is set to 0 (PIE disabled). The state of this bit can be modified after reset by writing to the PIECTRL register (address 0x0000 0CE0).

Using these bits and signals the possible vector table mappings are shown in Table 1-10.

Table 1-10. Interrupt Vector Table Mapping

Vector MAPS	Vectors Fetched From	Address Range	VMAP	M0M1MAP	ENPIE
M1 Vector(1)	M1 SARAM Block	0x000000 - 0x00003F	0	0	X
M0 Vector(1)	M0 SARAM Block	0x000000 - 0x00003F	0	1	X
BROM Vector	Boot ROM Block	0x3FFFC0 - 0x3FFFFF	1	X	0
PIE Vector	PIE Block	0x000D00 - 0x000DFF	1	X	1

(1) Vector map M0 and M1 Vector is a reserved mode only. On the 28x devices these are used as SARAM.

The M1 and M0 vector table mapping are reserved for TI testing only. When using other vector mappings, the M0 and M1 memory blocks are treated as SARAM blocks and can be used freely without any restrictions.

After a device reset operation, the vector table is mapped as shown in Table 1-11.

Copyright © 2012–2019, Texas Instruments Incorporated

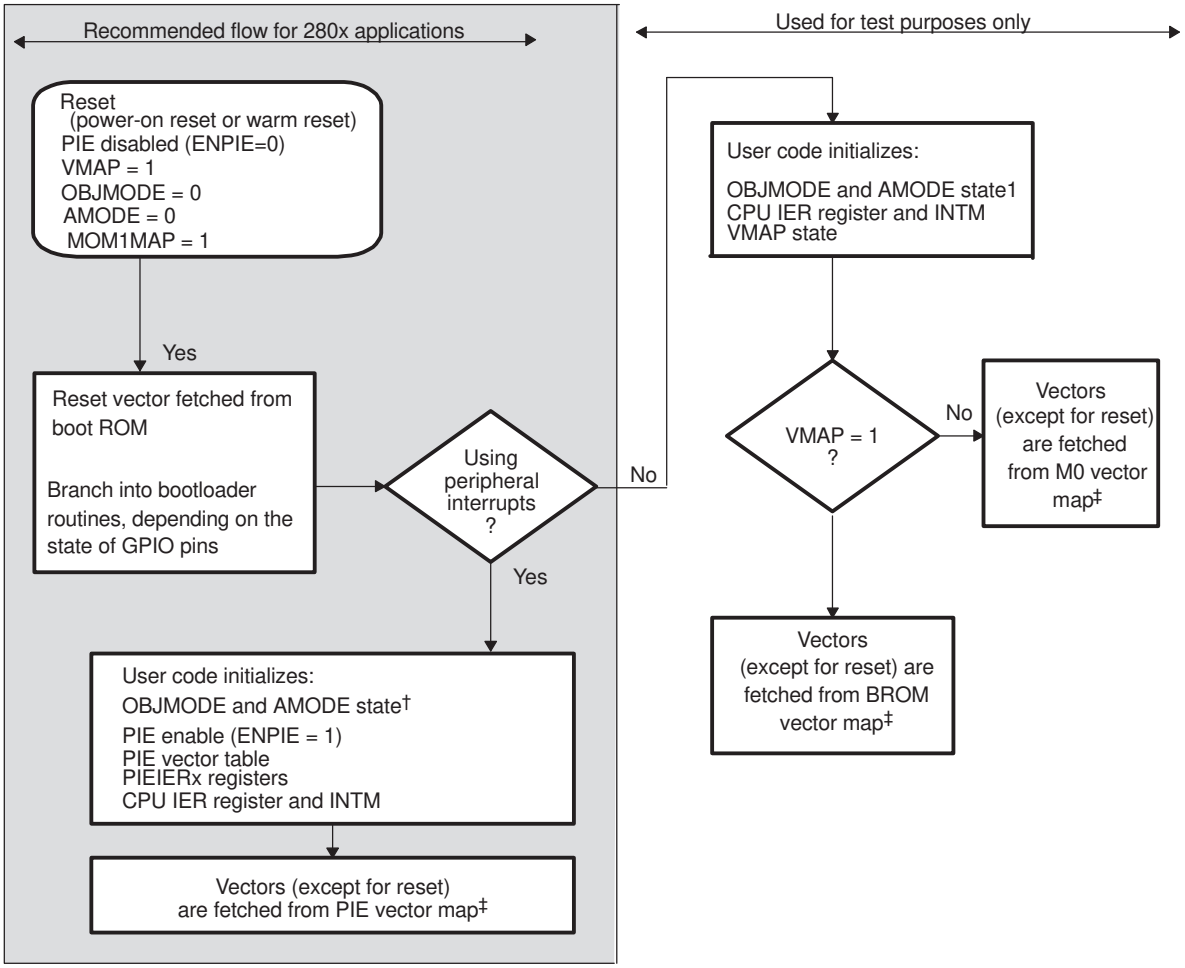
Table 1-11. Vector Table Mapping After Reset Operation

Vector MAPS	Reset Fetched From	Address Range	VMAP[1]	M0M1MAP[1]	ENPIE[1]
BROM Vector[2]	Boot ROM Block	0x3FFFC0 - 0x3FFFFF	1	1	0

(1) On the 28x devices, the VMAP and M0M1MAP modes are set to 1 on reset. The ENPIE mode is forced to 0 on reset.
(2) The reset vector is always fetched from the boot ROM.

After the reset and boot is complete, the PIE vector table should be initialized by the user's code. Then the application enables the PIE vector table. From that point on the interrupt vectors are fetched from the PIE vector table. Note: when a reset occurs, the reset vector is always fetched from the vector table as shown in Table 1-11. After a reset the PIE vector table is always disabled. Figure 1-6 illustrates the process by which the vector table mapping is selected.

Figure 1-6. Control Subsystem C28x Processor After Reset Flow



A The compatibility operating mode of the 28x CPU is determined by a combination of the OBJMODE and AMODE bits in Status Register 1 (ST1):

Operating Mode	OBJMODE	AMODE
C28x Mode	1	0
24x/240xA Source-Compatible	1	1
C27x Object-Compatible	0	0 (Default at reset)

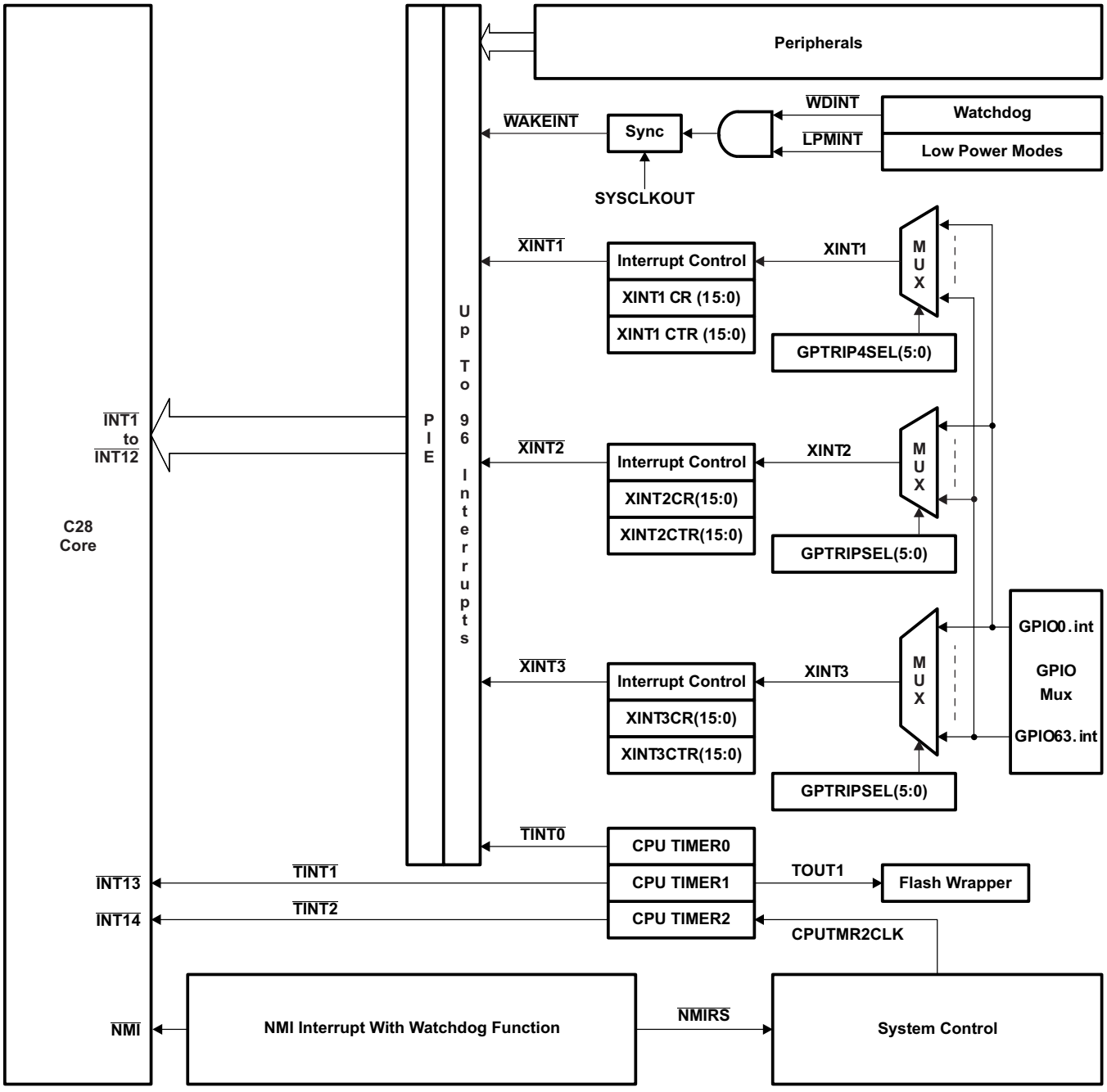
B The reset vector is always fetched from the boot ROM.

Copyright © 2012–2019, Texas Instruments Incorporated

1.5.4.3 Interrupt Sources

Figure 1-7 shows how the various interrupt sources are multiplexed within the devices. This multiplexing (MUX) scheme may not be exactly the same on all 28x devices. See the device data manual for details.

Figure 1-7. PIE Interrupt Sources and External Interrupts XINT1/XINT2/XINT3



1.5.4.3.1 Procedure for Handling Multiplexed Interrupts

The PIE module multiplexes eight peripheral and external pin interrupts into one CPU interrupt. These interrupts are divided into 12 groups: PIE group 1 - PIE group 12. Each group has an associated enable PIEIER and flag PIEIFR register. These registers are used to control the flow of interrupts to the CPU. The PIE module also uses the PIEIER and PIEIFR registers to decode to which interrupt service routine the CPU should branch.

Copyright © 2012–2019, Texas Instruments Incorporated

There are three steps that should be followed when clearing bits within the PIEIFR and the PIEIER registers:

Rule 1: Never clear a PIEIFR bit by software

An incoming interrupt may be lost while a write or a read-modify-write operation to the PIEIFR register takes place. To clear a PIEIFR bit, the pending interrupt must be serviced. If you want to clear the PIEIFR bit without executing the normal service routine, then use the following procedure:

1. Set the EALLOW bit to allow modification to the PIE vector table.
2. Modify the PIE vector table so that the vector for the peripheral's service routine points to a temporary ISR. This temporary ISR will only perform a return from interrupt (IRET) operation.
3. Enable the interrupt so that the interrupt will be serviced by the temporary ISR.
4. After the temporary interrupt routine is serviced, the PIEIFR bit will be clear
5. Modify the PIE vector table to re-map the peripheral's service routine to the proper service routine.
6. Clear the EALLOW bit.

Rule 2: Procedure for software-prioritizing interrupts

Use the method found in the device support examples in controlSUITE (literature number SPRCA85).

1. Use the CPU IER register as a global priority and the individual PIEIER registers for group priorities. In this case the PIEIER register is only modified within an interrupt. In addition, only the PIEIER for the same group as the interrupt being serviced is modified. This modification is done while the PIEACK bit holds additional interrupts back from the CPU.
2. Never disable a PIEIER bit for a group when servicing an interrupt from an unrelated group.

Rule 3: Disabling interrupts using PIEIER

If the PIEIER registers are used to enable and then later disable an interrupt then the procedure described in Section 1.5.4.3.2 must be followed.

1.5.4.3.2 Procedures for Enabling And Disabling Multiplexed Peripheral Interrupts

The proper procedure for enabling or disabling an interrupt is by using the peripheral interrupt enable/disable flags. The primary purpose of the PIEIER and CPU IER registers is for software prioritization of interrupts within the same PIE interrupt group. The software package *C280x C/C++ Header Files and Peripheral Examples in C* (literature number SPRC191) includes an example that illustrates this method of software prioritizing interrupts. Should bits within the PIEIER registers need to be cleared outside of this context, one of the following two procedures should be followed. The first method preserves the associated PIE flag register so that interrupts are not lost. The second method clears the associated PIE flag register.

Method 1: Use the PIEIERx register to disable the interrupt and preserve the associated PIEIFRx flags.

To clear bits within a PIEIERx register while preserving the associated flags in the PIEIFRx register, the following procedure should be followed:

1. Disable global interrupts (INTM = 1).
2. Clear the PIEIERx.y bit to disable the interrupt for a given peripheral. This can be done for one or more peripherals within the same group.
3. Wait five cycles. This delay is required to be sure that any interrupt that was incoming to the CPU has been flagged within the CPU IFR register.
4. Clear the CPU IFRx bit for the peripheral group. This is a safe operation on the CPU IFR register.
5. Clear the PIEACKx bit for the peripheral group.
6. Enable global interrupts (INTM = 0).

Method 2: Use the PIEIERx register to disable the interrupt and clear the associated PIEIFRx flags.

To perform a software reset of a peripheral interrupt and clear the associated flag in the PIEIFRx register and CPU IFR register, the following procedure should be followed:

1. Disable global interrupts (INTM = 1).
2. Set the EALLOW bit.

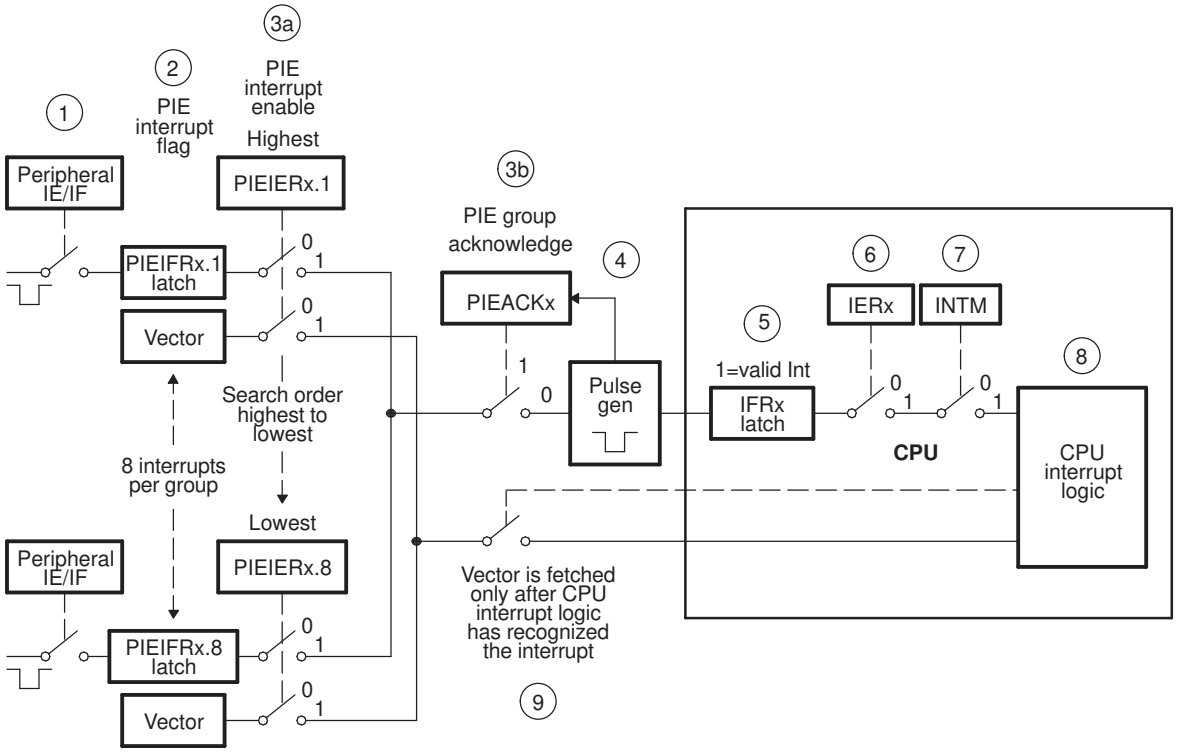
Copyright © 2012–2019, Texas Instruments Incorporated

3. Modify the PIE vector table to temporarily map the vector of the specific peripheral interrupt to a empty interrupt service routine (ISR). This empty ISR will only perform a return from interrupt (IRET) instruction. This is the safe way to clear a single PIEIFRx.y bit without losing any interrupts from other peripherals within the group.
4. Disable the peripheral interrupt at the peripheral register.
5. Enable global interrupts (INTM = 0).
6. Wait for any pending interrupt from the peripheral to be serviced by the empty ISR routine.
7. Disable global interrupts (INTM = 1).
8. Modify the PIE vector table to map the peripheral vector back to its original ISR.
9. Clear the EALLOW bit.
10. Disable the PIEIER bit for given peripheral.
11. Clear the IFR bit for given peripheral group (this is safe operation on CPU IFR register).
12. Clear the PIEACK bit for the PIE group.
13. Enable global interrupts.

1.5.4.3.3 Flow of a Multiplexed Interrupt Request From a Peripheral to the CPU

Figure 1-8 shows the flow with the steps shown in circled numbers. Following the diagram, the steps are described.

Figure 1-8. Multiplexed Interrupt Request Flow



1. Any peripheral or external interrupt within the PIE group generates an interrupt. If interrupts are enabled within the peripheral module then the interrupt request is sent to the PIE module.
2. The PIE module recognizes that interrupt y within PIE group x (INTx.y) has asserted an interrupt and the appropriate PIE interrupt flag bit is latched: PIEIFRx.y = 1.
3. For the interrupt request to be sent from the PIE to the CPU, both of the following conditions must be true:
   a. The proper enable bit must be set (PIEIERx.y = 1) and
   b. The PIEACKx bit for the group must be clear.

Copyright © 2012–2019, Texas Instruments Incorporated

4. If both conditions in 3a and 3b are true, then an interrupt request is sent to the CPU and the acknowledge bit is again set (PIEACKx = 1). The PIEACKx bit will remain set until you clear it to indicate that additional interrupts from the group can be sent from the PIE to the CPU.
5. The CPU interrupt flag bit is set (CPU IFRx = 1) to indicate a pending interrupt x at the CPU level.
6. If the CPU interrupt is enabled (CPU IER bit x = 1, or DBGIER bit x = 1) AND the global interrupt mask is clear (INTM = 0) then the CPU will service the INTx.
7. The CPU recognizes the interrupt and performs the automatic context save, clears the IER bit, sets INTM, and clears EALLOW. All of the steps that the CPU takes in order to prepare to service the interrupt are documented in the *TMS320C28x DSP CPU and Instruction Set Reference Guide* (literature number SPRU430).
8. The CPU will then request the appropriate vector from the PIE.
9. For multiplexed interrupts, the PIE module uses the current value in the PIEIERx and PIEIFRx registers to decode which vector address should be used. There are two possible cases:
   a. The vector for the highest priority interrupt within the group that is both enabled in the PIEIERx register and flagged as pending in the PIEIFRx is fetched and used as the branch address. In this manner if an even higher priority enabled interrupt was flagged after Step 7, it will be serviced first.
   b. If no flagged interrupts within the group are enabled, then the PIE will respond with the vector for the highest priority interrupt within that group. That is the branch address used for INTx.1. This behavior corresponds to the 28x TRAP or INT instructions.

NOTE: Because the PIEIERx register is used to determine which vector will be used for the branch, you must take care when clearing bits within the PIEIERx register. The proper procedure for clearing bits within a PIEIERx register is described and failure to follow these steps can result in changes occurring to the PIEIERx register after an interrupt has been passed to the CPU. In this case, the PIE will respond as if a TRAP or INT instruction was executed unless there are other interrupts both pending and enabled.

At this point, the PIEIFRx.y bit is cleared and the CPU branches to the vector of the interrupt fetched from the PIE

1.5.4.3.4 The PIE Vector Table

Table 1-13 consists of a 256 x 16 SARAM block that can also be used as RAM (in data space only) if the PIE block is not in use. The PIE vector table contents are undefined on reset. The CPU fixes interrupt priority for INT1 to INT12. The PIE controls priority for each group of eight interrupts. For example, if INT1.1 should occur simultaneously with INT8.1, both interrupts are presented to the CPU simultaneously by the PIE block, and the CPU services INT1.1 first. If INT1.1 should occur simultaneously with INT1.8, then INT1.1 is sent to the CPU first and then INT1.8 follows. Interrupt prioritization is performed during the vector fetch portion of the interrupt processing.

When the PIE is enabled, a TRAP #1 through TRAP #12 or an INTR INT1 to INTR INT12 instruction transfers program control to the interrupt service routine corresponding to the first vector within the PIE group. For example: TRAP #1 fetches the vector from INT1.1, TRAP #2 fetches the vector from INT2.1 and so forth. Similarly an OR IFR, #16-bit operation causes the vector to be fetched from INTR1.1 to INTR12.1 locations, if the respective interrupt flag is set. All other TRAP, INTR, OR IFR,#16-bit operations fetch the vector from the respective table location. The vector table is EALLOW protected.

Out of the 96 possible MUXed interrupts in Table 1-12, 66 interrupts are currently used. The remaining interrupts are reserved for future devices. These reserved interrupts can be used as software interrupts if they are enabled at the PIEIFRx level, provided none of the interrupts within the group is being used by a peripheral. Otherwise, interrupts coming from peripherals may be lost by accidentally clearing their flags when modifying the PIEIFR.

To summarize, there are two safe cases when the reserved interrupts can be used as software interrupts:

- No peripheral within the group is asserting interrupts.
- No peripheral interrupts are assigned to the group. For example, PIE group 11 and 12 do not have any peripherals attached to them.

Copyright © 2012–2019, Texas Instruments Incorporated

The interrupt grouping for peripherals and external interrupts connected to the PIE module is shown in Table 1-12. Each row in the table shows the eight interrupts multiplexed into a particular CPU interrupt.

Table 1-12. PIE Vector Table Mapping

	INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
INT1	C28.LPMWAKE (C28LPM) 0x0D4E	TINT0 (TIMER 0) 0x0D4C	Reserved ---- 0x0D4A	XINT2 ---- 0x0D48	XINT1 ---- 0x0D46	Reserved ---- 0x0D44	ADCINT2 (ADC) 0x0D42	ADCINT1 (ADC) 0x0D40
INT2	EPWM8_TZINT (ePWM8) 0x0D5E	EPWM7_TZINT (ePWM7) 0x0D5C	EPWM6_TZINT (ePWM6) 0x0D5A	EPWM5_TZINT (ePWM5) 0x0D58	EPWM4_TZINT (ePWM4) 0x0D56	EPWM3_TZINT (ePWM3) 0x0D54	EPWM2_TZINT (ePWM2) 0x0D52	EPWM1_TZINT (ePWM1) 0x0D50
INT3	EPWM8_INT (ePWM8) 0x0D6E	EPWM7_INT (ePWM7) 0x0D6C	EPWM6_INT (ePWM6) 0x0D6A	EPWM5_INT (ePWM5) 0x0D68	EPWM4_INT (ePWM4) 0x0D66	EPWM3_INT (ePWM3) 0x0D64	EPWM2_INT (ePWM2) 0x0D62	EPWM1_INT (ePWM1) 0x0D60
INT4	EPWM9_TZINT (ePWM9) 0x0D7E	Reserved ---- 0x0D7C	ECAP6_INT (eCAP6) 0x0D7A	ECAP5_INT (eCAP5) 0x0D78	ECAP4_INT (eCAP4) 0x0D76	ECAP3_INT (eCAP3) 0x0D74	ECAP2_INT (eCAP2) 0x0D72	ECAP1_INT (eCAP1) 0x0D70
INT5	EPWM9_INT (ePWM9) 0x0D8E	Reserved ---- 0x0D8C	Reserved ---- 0x0D8A	Reserved ---- 0x0D88	Reserved ---- 0x0D86	EQEP3_INT (eQEP3) 0x0D84	EQEP2_INT (eQEP2) 0x0D82	EQEP1_INT (eQEP1) 0x0D80
INT6	Reserved --- 0x0D9E	Reserved --- 0x0D9C	MXINTA (McBSP-A) 0x0D9A	MRINTA (McBSP-A) 0x0D98	Reserved --- 0x0D96	Reserved ---- 0x0D94	SPITXINTA (SPI-A) 0x0D92	SPIRXINTA (SPI-A) 0x0D90
INT7	Reserved ---- 0x0DAE	Reserved ---- 0x0DAC	DINTCH6 (C28 DMA) 0x0DAA	DINTCH5 (C28 DMA) 0x0DA8	DINTCH4 (C28 DMA) 0x0DA6	DINTCH3 (C28 DMA) 0x0DA4	DINTCH2 (C28 DMA) 0x0DA2	DINTCH1 (C28 DMA) 0x0DA0
INT8	Reserved ---- 0x0DBE	Reserved ---- 0x0DBC	Reserved ---- 0x0DBA	Reserved ---- 0x0DB8	Reserved ----- 0x0DB6	Reserved ---- 0x0DB4	I2CINT2A (I2C-A) 0x0DB2	I2CINT1A (I2C-A) 0x0DB0
INT9	Reserved ---- 0x0DCE	Reserved ---- 0x0DCC	Reserved ---- 0x0DCA	Reserved ---- 0x0DC8	Reserved ---- 0x0DC6	Reserved ---- 0x0DC4	SCITXINTA (SCI-A) 0x0DC2	SCIRXINTA (SCI-A) 0x0DC0
INT10	ADCINT8 (ADC) 0x0DDE	ADCINT7 (ADC) 0x0DDC	ADCINT6 (ADC) 0x0DDA	ADCINT5 (ADC) 0x0DD8	ADCINT4 (ADC) 0x0DD6	ADCINT3 (ADC) 0x0DD4	ADCINT2 (ADC) 0x0DD2	ADCINT1 (ADC) 0x0DD0
INT11	Reserved ---- 0x0DEE	Reserved ---- 0x0DEC	Reserved ---- 0x0DEA	Reserved ---- 0x0DE8	MTOCIPCINT4 (IPC) 0x0DE6	MTOCIPCINT3 (IPC) 0x0DE4	MTOCIPCINT2 (IPC) 0x0DE2	MTOCIPCINT1 (IPC) 0x0DE0
INT12	LUF (C28FPU) 0x0DFE	LVF (C28FPU) 0x0DFC	EPI ---- 0x0DFA	C28RAMACCVIOL (Memory) 0x0DF8	C28RAMSINGERR (Memory) 0x0DF6	Reserved ---- 0x0DF4	C28FLSINGERR (Memory) 0x0DF2	XINT3 (Ext. Int. 3) 0x0DF0

Table 1-13. PIE Vector Table

Name	VECTOR ID	Address[1]	Size(x16)	Description[2]	CPU Priority	PIE Group Priority
Reset	0	0x00000D00	2	Reset is always fetched from location 0x003F FFC0 in BootROM	1 (highest)	-
INT1	1	0x00000D02	2	Not used. See PIE Group 1	5	-
INT2	2	0x00000D04	2	Not used. See PIE Group 2	6	-
INT3	3	0x00000D06	2	Not used. See PIE Group 3	7	-

(1) Reset is always fetched from location 0x003F FFC0 in Boot ROM.
(2) All the locations within the PIE vector table are EALLOW protected.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-13. PIE Vector Table (continued)

Name	VECTOR ID	Address[1]	Size(x16)	Description[2]	CPU Priority	PIE Group Priority
INT4	4	0x00000D08	2	Not used. See PIE Group 4	8	-
INT5	5	0x00000D0A	2	Not used. See PIE Group 5	9	-
INT6	6	0x00000D0C	2	Not used. See PIE Group 6	10	-
INT7	7	0x00000D0E	2	Not used. See PIE Group 7	11	-
INT8	8	0x00000D10	2	Not used. See PIE Group 8	12	-
INT9	9	0x00000D12	2	Not used. See PIE Group 9	13	-
INT10	10	0x00000D14	2	Not used. See PIE Group 10	14	-
INT11	11	0x00000D16	2	Not used. See PIE Group 11	15	-
INT12	12	0x00000D18	2	Not used. See PIE Group 12	16	-
INT13	13	0x00000D1A	2	CPU-Timer1	17	-
INT14	14	0x00000D1C	2	CPU-Timer2 (for TI/RTOS use)	18	-
DATALOG	15	0x00000D1E	2	CPU Data Logging Interrupt	19 (lowest)	-
RTOSINT	16	0x0000 0D20	2	CPU Real-Time OS Interrupt	4	--
EMUINT	17	0x0000 0D22	2	CPU Emulation Interrupt	2	-
NMI	18	0x0000 0D24	2	External Non-Maskable Interrupt	3	-
ILLEGAL	19	0x0000 0D26	2	Illegal Operation	--	-
USER1	20	0x0000 0D28	2	User-Defined Trap	--	-
USER2	21	0x0000 0D2A	2	User Defined Trap	--	-
USER3	22	0x0000 0D2C	2	User Defined Trap	--	-
USER4	23	0x0000 0D2E	2	User Defined Trap	--	--
USER5	24	0x0000 0D30	2	User Defined Trap	--	-
USER6	25	0x0000 0D32	2	User Defined Trap	--	-
USER7	26	0x0000 0D34	2	User Defined Trap	--	-
USER8	27	0x0000 0D36	2	User Defined Trap	--	-
USER9	28	0x0000 0D38	2	User Defined Trap	--	-
USER10	29	0x0000 0D3A	2	User Defined Trap	--	-
USER11	30	0x0000 0D3C	2	User Defined Trap	--	-
USER12	31	0x0000 0D3E	2	User Defined Trap	--	-
PIE Group 1 Vectors -MUXed into CPU INT1						
INT1.1	32	0x0000 0D40	2	ADCINT1 (ADC)	5	1 (highest)
INT1.2	33	0x0000 0D42	2	ADCINT2 (ADC)	5	2
INT1.3	34	0x0000 0D44	2	Reserved	5	3
INT1.4	35	0x0000 0D46	2	XINT1	5	4
INT1.5	36	0x0000 0D48	2	XINT2	5	5
INT1.6	37	0x0000 0D4A	2	Reserved	5	6
INT1.7	38	0x0000 0D4C	2	TINT0 (C28x Timer 0)	5	7

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-13. PIE Vector Table (continued)

Name	VECTOR ID	Address[1]	Size(x16)	Description[2]	CPU Priority	PIE Group Priority
INT1.8	39	0x0000 0D4E	2	WAKEINT (C28 LPM)	5	8 (lowest)
PIE Group 2 Vectors -MUXed into CPU INT2						
INT2.1	40	0x0000 0D50	2	EPWM1_TZINT	6	1 (highest)
INT2.2	41	0x0000 0D52	2	EPWM2_TZINT	6	2
INT2.3	42	0x0000 0D54	2	EPWM3_TZINT	6	3
INT2.4	43	0x0000 0D56	2	EPWM4_TZINT	6	4
INT2.5	44	0x0000 0D58	2	EPWM5_TZINT	6	5
INT2.6	45	0x0000 0D5A	2	EPWM6_TZINT	6	6
INT2.7	46	0x0000 0D5C	2	EPWM7_TZINT	6	7
INT2.8	47	0x0000 0D5E	2	EPWM8_TZINT	6	8 (lowest)
PIE Group 3 Vectors -MUXed into CPU INT3						
INT3.1	48	0x0000 0D60	2	EPWM1_INT	7	1 (highest)
INT3.2	49	0x0000 0D62	2	EPWM2_INT	7	2
INT3.3	50	0x0000 0D64	2	EPWM3_INT	7	3
INT3.4	51	0x0000 0D66	2	EPWM4_INT	7	4
INT3.5	52	0x0000 0D68	2	EPWM5_INT	7	5
INT3.6	53	0x0000 0D6A	2	EPWM6_INT	7	6
INT3.7	54	0x0000 0D6C	2	EPWM7_INT	7	7
INT3.8	55	0x0000 0D6E	2	EPWM8_INT	7	8 (lowest)
PIE Group 4 Vectors -MUXed into CPU INT4						
INT4.1	56	0x0000 0D70	2	ECAP1_INT	8	1 (highest)
INT4.2	57	0x0000 0D72	2	ECAP2_INT	8	2
INT4.3	58	0x0000 0D74	2	ECAP3_INT	8	3
INT4.4	58	0x0000 0D76	2	ECAP4_INT	8	4
INT4.5	60	0x0000 0D78	2	ECAP5_INT	8	5
INT4.6	61	0x0000 0D7A	2	ECAP6_INT	8	6
INT4.7	62	0x0000 0D7C	2	Reserved	8	7
INT4.8	63	0x0000 0D7E	2	EPWM9_TZINT	8	8 (lowest)
PIE Group 5 Vectors -MUXed into CPU INT5						
INT5.1	64	0x0000 0D80	2	EQEP1_INT	9	1 (highest)
INT5.2	65	0x0000 0D82	2	EQEP2_INT	9	2
INT5.3	66	0x0000 0D84	2	EQEP3_INT	9	3
INT5.4	67	0x0000 0D86	2	Reserved	9	4
INT5.5	68	0x0000 0D88	2	Reserved	9	5
INT5.6	69	0x0000 0D8A	2	Reserved	9	6
INT5.7	70	0x0000 0D8C	2	Reserved	9	7
INT5.8	71	0x0000 0D8E	2	EPWM9_INT	9	8 (lowest)
PIE Group 6 Vectors -MUXed into CPU INT6						
INT6.1	72	0x0000 0D90	2	SPIRXINTA (SPI-A)	10	1 (highest)
INT6.2	73	0x0000 0D92	2	SPITXINTA (SPI-A)	10	2
INT6.3	74	0x0000 0D94	2	Reserved	10	3
INT6.4	75	0x0000 0D96	2	Reserved	10	4
INT6.5	76	0x0000 0D98	2	MRINTA (McBSP-A)	10	5

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-13. PIE Vector Table (continued)

Name	VECTOR ID	Address[1]	Size(x16)	Description[2]	CPU Priority	PIE Group Priority
INT6.6	77	0x0000 0D9A	2	MXINTA (McBSP-A)	10	6
INT6.7	78	0x0000 0D9C	2	Reserved	10	7
INT6.8	79	0x0000 0D9E	2	Reserved	10	8 (lowest)
PIE Group 7 Vectors -MUXed into CPU INT7						
INT7.1	80	0x0000 0DA0	2	DINTCH1 (C28 DMA)	11	1 (highest)
INT7.2	81	0x0000 0DA2	2	DINTCH2 (C28 DMA)	11	2
INT7.3	82	0x0000 0DA4	2	DINTCH3 (C28 DMA)	11	3
INT7.4	83	0x0000 0DA6	2	DINTCH4 (C28 DMA)	11	4
INT7.5	84	0x0000 0DA8	2	DINTCH5 (C28 DMA)	11	5
INT7.6	85	0x0000 0DAA	2	DINTCH6 (C28 DMA)	11	6
INT7.7	86	0x0000 0DAC	2	Reserved	11	7
INT7.8	87	0x0000 0DAE	2	Reserved	11	8 (lowest)
PIE Group 8 Vectors -MUXed into CPU INT8						
INT8.1	88	0x0000 0DB0	2	I2CINT2-A (I2C-A)	12	1 (highest)
INT8.2	89	0x0000 0DB2	2	I2CINT2-A (I2C-A)	12	2
INT8.3	90	0x0000 0DB4	2	Reserved	12	3
INT8.4	91	0x0000 0DB6	2	Reserved	12	4
INT8.5	92	0x0000 0DB8	2	Reserved	12	5
INT8.6	93	0x0000 0DBA	2	Reserved	12	6
INT8.7	94	0x0000 0DBC	2	Reserved	12	7
INT8.8	95	0x0000 0DBE	2	Reserved	12	8 (lowest)
PIE Group 9 Vectors -MUXed into CPU INT9						
INT9.1	96	0x0000 0DC0	2	SCIRXINTA	13	1 (highest)
INT9.2	97	0x0000 0DC2	2	SCITXINTA	13	2
INT9.3	98	0x0000 0DC4	2	Reserved	13	3
INT9.4	99	0x0000 0DC6	2	Reserved	13	4
INT9.5	100	0x0000 0DC8	2	Reserved	13	5
INT9.6	101	0x0000 0DCA	2	Reserved	13	6
INT9.7	102	0x0000 0DCC	2	Reserved	13	7
INT9.8	103	0x0000 0DCE	2	Reserved	13	8 (lowest)
PIE Group 10 Vectors -MUXed into CPU INT10						
INT10.1	104	0x0000 0DD0	2	ADCINT1(ADC)	14	1 (highest)
INT10.2	105	0x0000 0DD2	2	ADCINT2(ADC)	14	2
INT10.3	106	0x0000 0DD4	2	ADCINT3(ADC)	14	3
INT10.4	107	0x0000 0DD6	2	ADCINT4(ADC)	14	4
INT10.5	108	0x0000 0DD8	2	ADCINT5(ADC)	14	5
INT10.6	109	0x0000 0DDA	2	ADCINT6(ADC)	14	6
INT10.7	110	0x0000 0DDC	2	ADCINT7(ADC)	14	7
INT10.8	111	0x0000 0DDE	2	ADCINT8(ADC)	14	8 (lowest)
PIE Group 11 Vectors -MUXed into CPU INT11						
INT11.1	112	0x0000 0DE0	2	MTOCIPCINT1	15	1 (highest)
INT11.2	113	0x0000 0DE2	2	MTOCIPCINT2	15	2

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-13. PIE Vector Table (continued)

Name	VECTOR ID	Address[1]	Size(x16)	Description[2]	CPU Priority	PIE Group Priority
INT11.3	114	0x0000 0DE4	2	MTOCIPCINT3	15	3
INT11.4	115	0x0000 0DE6	2	MTOCIPCINT4	15	4
INT11.5	116	0x0000 0DE8	2	Reserved	15	5
INT11.6	117	0x0000 0DEA	2	Reserved	15	6
INT11.7	118	0x0000 0DEC	2	Reserved	15	7
INT11.8	119	0x0000 0DEE	2	Reserved	15	8 (lowest)
PIE Group 12 Vectors -MUXed into CPU INT12						
INT12.1	120	0x0000 0DF0	2	XINT3	16	1 (highest)
INT12.2	121	0x0000 0DF2	2	C28FLSINGERR	16	2
INT12.3	122	0x0000 0DF4	2	Reserved	16	3
INT12.4	123	0x0000 0DF6	2	C28RAMSINGERR	16	4
INT12.5	124	0x0000 0DF8	2	C28RAMACCVIOL	16	5
INT12.6	125	0x0000 0DFA	2	Reserved	16	6
INT12.7	126	0x0000 0DFC	2	LVF(C28FPU)	16	7
INT12.8	127	0x0000 0DFE	2	LUF(C28FPU)	16	8 (lowest)

1.5.4.4 PIE Registers

The PIE registers are as follows:

- PIE Configuration Register (PIECTRL)
- PIE Interrupt Acknowledgment Register (PIEACK)
- PIE Interrupt Flag Registers (PIEIFR). There are 12 PIEIFR registers, one for each CPU interrupt in the PIE module (INT1-INT12).
- PIE Interrupt Enable Registers (PIEIER). There are 12 PIEIER () one for each CPU interrupt in the PIE module (INT1-INT12).

Please refer to the respective PIE registers for details.

1.5.4.4.1 CPU Interrupt Flag Register (IFR)

The CPU interrupt flag register (IFR), is a 16-bit, CPU register and is used to identify and clear pending interrupts. The IFR contains flag bits for all the maskable interrupts at the CPU level (INT1-INT14, DLOGINT and RTOSINT). When the PIE is enabled, the PIE module multiplexes interrupt sources for INT1-INT12.

When a maskable interrupt is requested, the flag bit in the corresponding peripheral control register is set to 1. If the corresponding mask bit is also 1, the interrupt request is sent to the CPU, setting the corresponding flag in the IFR. This indicates that the interrupt is pending or waiting for acknowledgment.

To identify pending interrupts, use the PUSH IFR instruction and then test the value on the stack. Use the OR IFR instruction to set IFR bits and use the AND IFR instruction to manually clear pending interrupts. All pending interrupts are cleared with the AND IFR #0 instruction or by a hardware reset.

The following events also clear an IFR flag:

- The CPU acknowledges the interrupt.
- The 28x device is reset.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE:

1. To clear a CPU IFR bit, you must write a zero to it, not a one.
2. When a maskable interrupt is acknowledged, only the IFR bit is cleared automatically. The flag bit in the corresponding peripheral control register is not cleared. If an application requires that the control register flag be cleared, the bit must be cleared by software.
3. When an interrupt is requested by an INTR instruction and the corresponding IFR bit is set, the CPU does not clear the bit automatically. If an application requires that the IFR bit be cleared, the bit must be cleared by software.
4. IMR and IFR registers pertain to core-level interrupts. All peripherals have their own interrupt mask and flag bits in their respective control/configuration registers. Note that several peripheral interrupts are grouped under one core-level interrupt.

1.5.4.4.2 Interrupt Enable Register (IER) and Debug Interrupt Enable Register (DBGIER)

The IER 16-bit CPU register contains enable bits for all the maskable CPU interrupt levels (INT1-INT14, RTOSINT and DLOGINT). Neither NMI nor $\overline{\text{XRS}}$ is included in the IER; thus, IER has no effect on these interrupts.

You can read the IER to identify enabled or disabled interrupt levels, and you can write to the IER to enable or disable interrupt levels. To enable an interrupt level, set its corresponding IER bit to one using the OR IER instruction. To disable an interrupt level, set its corresponding IER bit to zero using the AND IER instruction. When an interrupt is disabled, it is not acknowledged, regardless of the value of the INTM bit. When an interrupt is enabled, it is acknowledged if the corresponding IFR bit is one and the INTM bit is zero.

When using the OR IER and AND IER instructions to modify IER bits make sure they do not modify the state of bit 15 (RTOSINT) unless a real-time operating system is present.

When a hardware interrupt is serviced or an INTR instruction is executed, the corresponding IER bit is cleared automatically. When an interrupt is requested by the TRAP instruction the IER bit is not cleared automatically. In the case of the TRAP instruction if the bit needs to be cleared it must be done by the interrupt service routine.

At reset, all the IER bits are cleared to 0, disabling all maskable CPU level interrupts.

The Debug Interrupt Enable Register (DBGIER) is used only when the CPU is halted in real-time emulation mode. An interrupt enabled in the DBGIER is defined as a time-critical interrupt. When the CPU is halted in real-time mode, the only interrupts that are serviced are time-critical interrupts that are also enabled in the IER. If the CPU is running in real-time emulation mode, the standard interrupt-handling process is used and the DBGIER is ignored.

As with the IER, you can read the DBGIER to identify enabled or disabled interrupts and write to the DBGIER to enable or disable interrupts. To enable an interrupt, set its corresponding bit to 1. To disable an interrupt, set its corresponding bit to 0. Use the PUSH DBGIER instruction to read from the DBGIER and POP DBGIER to write to the DBGIER register. At reset, all the DBGIER bits are set to 0.

1.5.4.5 External Interrupt Control Registers

Three external interrupts, XINT1 - XINT3 are supported. Each of these external interrupts can be selected for negative or positive edge triggering and can also be enabled or disabled. The masked interrupts also contain a 16-bit free running up counter that is reset to zero when a valid interrupt edge is detected. This counter can be used to accurately time stamp the interrupt.

For XINT1/XINT2/XINT3, there is also a 16-bit counter that is reset to 0x000 whenever an interrupt edge is detected. These counters can be used to accurately time stamp an occurrence of the interrupt.

Copyright © 2012–2019, Texas Instruments Incorporated

1.5.5 Control Subsystem Exceptions Handling

The ITRAP (illegal exception trap condition) is the only type of exception (other than programmable interrupts; TRAPs and NMI), which the control subsystem supports. An ITRAP exception occurs if an illegal instruction is fetched and executed. On an ITRAP exception when the PIE is enabled, the program counter is loaded with the vector address located at address location 0x00000D26. When the PIE is disabled the vector address loaded in the program counter is fetched from the ROM Vector table location 0x3FFFE4.

NOTE: RAM Access Violation Interrupt (INT12.5):

PIE INT12.5 is defined for C28x RAM access violations, but for a **fetch access violation**, the C28x will receive zero data and will ITRAP. The ITRAP occurs in the same cycle at which the INT12.5 is fired to the C28x. Since the ITRAP has priority over normal interrupt, the ITRAP ISR gets executed before the INT12.5 ISR. The user needs to take this into account when servicing the ITRAP.

Please refer to the *Boot ROM* chapter for details on how the boot ROM handles this exception if it occurs during its execution.

1.5.6 Control Subsystem NMI (CNMI) Module

Similar to the MNMI module on the master subsystem, the control subsystem supports an NMI module that enables it to detect serious errors that can occur during the operation of the device. Possible errors that can trigger an NMI to the control subsystem are:

- CLOCKFAIL
- ACIBERR
- RAMUNCERR
- FLASHUNCERR

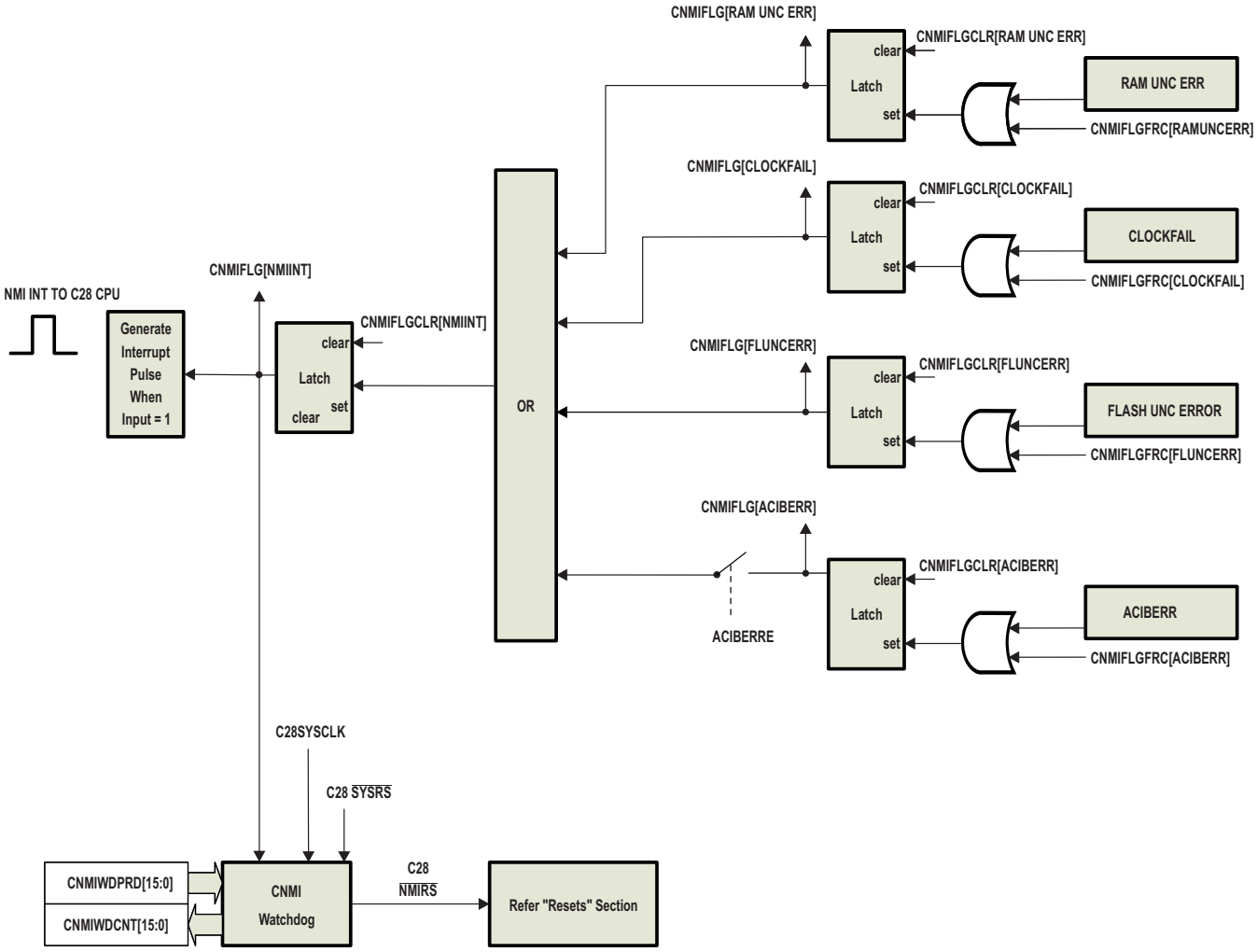
An NMI is generated to the control subsystem CPU if any of the above errors occur. If an NMI is triggered to the control subsystem, the NMI WatchDog timer will start counting and will reset the control subsystem if the counter value reaches a programmed period value. An NMI is generated to the Master CPU if the control subsystem is reset by the control subsystem watchdog, CNsMIWD.

The C28x NMI block can be accessed via the C28x NMI configuration registers (including the CNMIFLG, CNMIFLGCLR, and CNMIFLGFRC registers) to examine flag bits for the NMI sources, clear the flags, and force the flags to active state, respectively.

Figure 1-9 explains how an NMI is generated to the control subsystem CPU.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-9. Control Subsystem NMI Sources and CNMIWD



As shown in Figure 1-9, any of the listed errors can trigger an NMI to the C28x CPU There is also an NMI Flag Force (CNMIFLGFRC) register to simulate an NMI error condition to aid in debug and development. When an NMI error event occurs:

- Respective bit in the CNMIFLG register is set
- NMIINT bit in the CNMIFLG register is set
- CNMIWD timer is triggered and starts counting

The user has to handle the error condition that triggered the NMI by checking the individual flag bits in the CNMIFLG register, clearing the set bits in the CNMIFLG register, and clearing the NMIINT bit in the CNMIFLG register. Clearing the NMIINT bit in the CNMIFLG register will stop and reset the CNMIWD counter back to zero. If the NMIINT bit is not cleared by the time the CNMIWD counter reaches the value programmed in the CNMIWDPRD register, the control subsystem is reset and an NMI is triggered to the master subsystem NMI block.

1.5.6.1 Control Subsystem NMI Sources

This section explains the error events that can generate an NMI to the control subsystem CPU.

Copyright © 2012–2019, Texas Instruments Incorporated

1.5.6.1.1 Clock Fail Condition

A main oscillator verification circuit is provided to generate an error condition if the oscillator is running too fast or too slow or goes missing. This logic is referred to as missing clock detection logic. When a missing clock error is generated, the CLOCKFAIL bit (bit 1) of the CNMIFLG register is set, the clock source is switched to the 10 MHz internal oscillator, and the PLL is bypassed.

The CLOCKFAIL NMI is triggered to both the master and control subsystems. Since this NMI is enabled by default on power up, it is necessary for boot ROM to handle it. Refer to the *Boot ROM* chapter of this document for more details on how C-Boot ROM handles this NMI.

1.5.6.1.2 ACIBERR NMI

An NMI is generated to both the master subsystem and control subsystem when the error condition in the ACIB occurs. The ACIBERR bit in the CNMIFLG register is set, and to clear this NMI, the user has to clear the ACIBERR bit in the CNMIFLG register.

This NMI is disabled by default on reset, and the user can enable this error to trigger an NMI to the C28x CPU by setting the ACIBERRE bit in the CNMICFG register.

1.5.6.1.3 RAMUNCERR NMI

For information on the RAM Uncorrectable Error NMI, please refer to the *Internal Memory* chapter for more details. This NMI is generated when a double bit error is detected by the memory wrapper logic.

1.5.6.1.4 FLUNCERR NMI

For information on the FLASH Uncorrectable Error NMI, please refer to the *Internal Memory* chapter for more details.

1.5.6.2 Control Subsystem NMIWD (CNMIWD) Module

The control subsystem is equipped with an NMI Watchdog module that when a non-maskable interrupt is triggered, allocates time for the user software to handle the NMI by clearing the error conditions and clearing the respective flags in the CNMIFLG register or by acknowledging the NMI and gracefully shutting down the system. If none of the actions mentioned are taken, the CNMIWD counter keeps counting. As soon as the counter value reaches the CNMIWD period register value, it will generate a CNMIWD reset, which will reset the entire device.

The CNMIWD counter is clocked by the C28 system clock. The CNMI WatchDog Period register (CNMIWDPRD) can be programmed with a period limit as per user requirements with the clock cycle limit the user requires for software to handle or acknowledge the NMI.

1.5.6.2.1 Emulation Considerations

The CNMI watchdog module behaves as below under various debug conditions.

CPU Suspended	When the CPU is suspended, the NMI watchdog counter will be suspended.
Run-Free Mode	When the CPU is placed in run-free mode, the NMI watchdog counter will resume operation as normal.
Real-Time Single-Step Mode	When the CPU is in real-time single-step mode, the NMI watchdog counter will be suspended. The counter remains suspended even within real-time interrupts.
Real-Time Run-Free Mode	When the CPU is in real-time run-free mode, the NMI watchdog counter operates as normal.

Copyright © 2012–2019, Texas Instruments Incorporated

1.5.6.3 Handling of CNMI

User software must clear all the flag bits which are set in the CNMIFLG register before clearing the NMIINT bit (bit 0) of the CNMIFLG register. If the user clears the NMIINT bit of the CNMIFLG register before clearing all the individual flag bits, as soon as the NMIINT bit is cleared, it will be set back to "1" again. If a user returns from the NMI handler without clearing the NMIINT bit of the CNMIFLG register, then a reset is generated to the control subsystem when the CNMIWD counter value reaches the programmed period value.

1.6 Safety Features

This section gives details on different modules or features that safe guard device operation during run time and help catch serious errors that can occur. Also included are details on hardware behavior when any of the serious errors occur in the device.

1.6.1 Write Protection on Registers

1.6.1.1 Master Subsystem Write Protection

1.6.1.1.1 MWRALLOW

To meet the requirements of safety-critical applications and to safeguard system control registers from run-away software code in case of software design errors, hardware protection is provided to prevent contents of critical registers from getting corrupted.

Write protection to critical registers is achieved by using a "double write" method. In this method there is a MWRALLOW register, which if written with a value of 0xA5A5 A5A5, allows writes to all other "PROTECTED" registers defined in this specification. The MWRALLOW register is only writable in M3 privileged mode.

1.6.1.1.2 MLOCK

The MLOCK register is required to prevent accidental writes to the MSxMSEL shared memory (S0 to S7) RAM block mapping register.

This register lets users lock the configuration of Shared RAMs once configured by the master subsystem application. This register is a write once register and once written, it cannot be cleared or re-written until the respective reset that resets this register is generated. Refer to the *Internal Memory* chapter of this document for more details on shared RAM configuration registers.

1.6.1.2 Control Subsystem Write Protection

Several control registers are protected from spurious CPU writes by the EALLOW protection mechanism. The EALLOW bit in status register 1 (ST1) indicates the state of protection as shown in Table 1-14.

Table 1-14. Access to EALLOW-Protected Registers

EALLOW Bit	CPU Writes	CPU Reads	JTAG Writes	JTAG Reads
0	Ignored	Allowed	Allowed(1)	Allowed
1	Allowed	Allowed	Allowed	Allowed

(1) The EALLOW bit is overridden via the JTAG port, allowing full access of protected registers during debug from the Code Composer Studio interface.

At reset, the EALLOW bit is cleared, enabling EALLOW protection. While protected, all writes to protected registers by the CPU are ignored and only CPU reads, JTAG reads, and JTAG writes are allowed. If this bit is set, by executing the EALLOW instruction, the CPU is allowed to write freely to protected registers. After modifying registers, they can once again be protected by executing the EDI instruction to clear the EALLOW bit.

Copyright © 2012–2019, Texas Instruments Incorporated

1.6.1.3 Analog Subsystem Clocking Control Protection with Control System Lock Register (CLOCK)

As will be explained in the control subsystem clocking section, the ACIB and analog subsystem clocking can be configured by setting the divider in the CCLKCTL register. The control subsystem can choose to lock this clock configuration by setting the CCLKCTL bit (bit 1) in the CLOCK register. Once the control subsystem locks this register, it cannot be re-written until the analog subsystem and ACIB are reset. Please refer to Section 1.3 for more details on the resets that can reset the ACIB and the analog subsystem.

1.6.2 Missing Clock Detection Logic

A main oscillator verification circuit is provided that generates an error condition if the oscillator is running too fast or too slow or goes missing. This logic is referred to as Missing Clock Detection logic.

The missing clock circuit detects if the clock is missing on the X1 pin. In the event of a missing clock detection, the clock is automatically switched to the internal 10 MHz oscillator clock. This 10 MHz oscillator will not be fed to the PLL. It is to be used only for error handling on clock missing condition detection.

The following includes the missing clock detection scheme:

1. Missing clock detection circuitry is enabled by default so that a clock missing condition at power up can be detected.
2. Missing clock detection can be disabled by a user programmable register.
3. At power up, on the rising edge of $\overline{\text{XRS}}$, all counters related to the circuit are cleared (Described below)
4. There is a 3-bit counter which runs off the 10 MHz clock. This counter will overflow approximately every 800ns
5. There is also an 8-bit counter running with the OSCCLK source, which keeps updating a reference clock count register.
6. There are two programmable registers which contain programmable limits for missing clock condition detection: device reference clock count limit low, and clock count limit high registers. These registers are programmable to allow support for the entire range of reference clock frequencies possible from 4 to 100 MHz. At reset, they are initialized to support the entire range of clock input frequencies. Refer to the CLKLIMIT register description for reset values.
7. When the 3-bit 10 MHz clock counter overflows, it reads the OSCCLK reference clock count register value (MCLKSTS.REFCLKCNT) and checks if the value falls between the limits defined in the clock count limit registers
    1. If yes,
        a. then it means that the device input reference clock is still ticking within the defined frequency range, enabling the counter to count properly as expected
        b. since the reference clock is present, the OSCLK reference clock counter, count register (MCLKSTS.REFCLKCNT), and 10 MHz clock counter are cleared and both the counters in the circuit start counting again from 0.
    2. If no,
        a. the circuit will switch clocks to the 10 MHz internal oscillator clock - the system PLL is bypassed and the 10 MHz clock becomes the clock source to the device.
        b. since the reference clock is missing, the circuit generates a clock fail NMI to the M3 CPU and C28 CPU, if enabled by the MCLKEN.MCLKNMIEN bit
        c. the control subsystem PWM's, if enabled, are tripped automatically on clock fail.
8. On both the master and control subsystems, there is an NMI watch-dog, which is triggered when the missing clock condition is detected.
9. If the M3 CPU does not respond to the NMI interrupt by the time the NMI WD expires, a reset will be issued to the device.
10. On the control subsystem, upon receiving the missing clock NMI, the C28 CPU does the required error handling and enters standby mode in the software ISR for the NMI. If the CNMIWD expires it will generate a reset to the C28 CPU and C28 subsystem.

Copyright © 2012–2019, Texas Instruments Incorporated

11. On the control subsystem, the $\overline{\text{CLOCKFAIL}}$ signal is also fed to TZ5n trip signal of all the PWM modules. This allows the PWM outputs to trip in case of clock failure.
12. When a missing clock condition is detected, the counters are disabled. They need to be re-enabled by software after all error handling and when a proper clock source is available again

NOTE: **[1]**On a clock fail condition, nothing is done in hardware to enter standby mode on the C28 side. When a CLOCKFAIL NMI is generated, the C28 application software should configure the LPMCR registers to enter standby mode.

[2]Changing the clock limit values (REFCLKLO and HI limit) should be done after disabling the missing clock NMI, using the MCLKNMIEN bit in the MCLKEN register. After the registers are programmed for appropriate input clock frequencies, the MCLKNMIEN bit should be set again after at least a 900ns delay.

[3]At power up, the circuit only checks whether the clock is ticking or not because the limits encompass the entire range of clock input frequencies allowed in the device.

Example of clock limit register configuration: Internal Oscillator is at 10 MHz; 3-bit 10 MHz clock counter overflows every 800ns.

Table 1-15. Reference Clock Limits for Detecting a Missing Clock

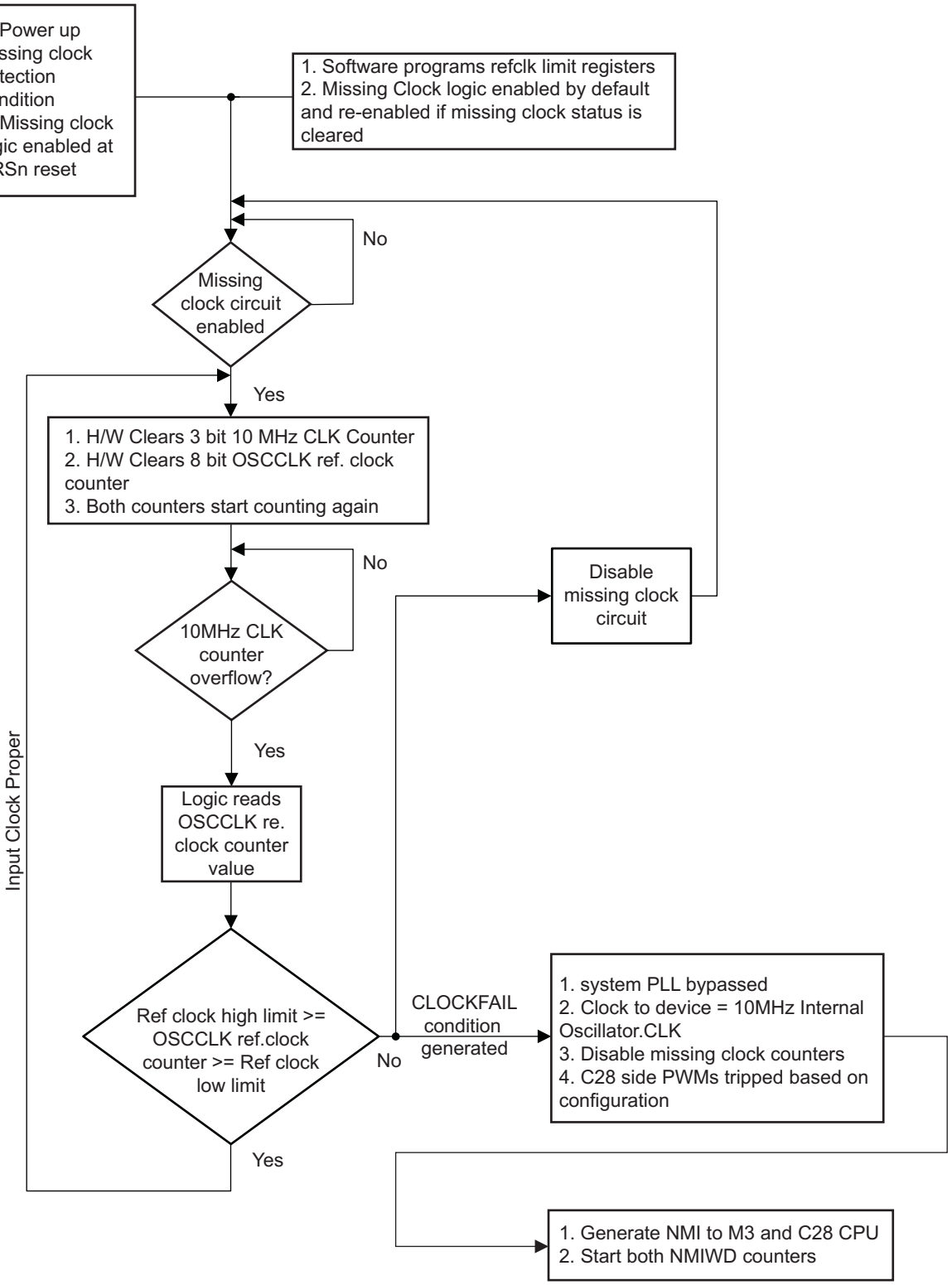
Ref Clock Frequency	REFCLKLOLIMIT	REFCLKHILIMIT
4 MHz (250ns)	0x3	0x4
10 MHz (100ns)	0x7	0x9
20 MHz (50ns)	0xF	0x11
100 MHz (10ns)	0x4E	0x52

Copyright © 2012–2019, Texas Instruments Incorporated

Note: The missing clock circuit is not active during the PLL 1024 cycle lock time. To avoid missing clock detection, the user should extend the external reset ($\overline{XRS}$) so it covers appropriate lock time.

Figure 1-10 shows the missing clock logic functional flow.

Figure 1-10. Missing Clock Detection Logic



Copyright © 2012–2019, Texas Instruments Incorporated

1.6.3 PLLSLIP Detection

On the master subsystem, an interrupt is generated (if enabled) to the NVIC when either the SYSTEM PLL or the USB PLL goes out of lock when previously locked.

In general, a PLL will go out of lock when the input clock to the PLL fluctuates if there is EMI interference on the device or if the input clock to PLL goes missing.

NVIC Interrupt No. 89 (System/USB PLL Out-of-Lock) is generated whenever the system PLL or USB PLL goes out of lock. The interrupt handler should check the status bits to find out which PLL has gone out of lock. It is the responsibility of the master subsystem application to indicate out-of-lock status to the control subsystem using an IPC. This has to be taken care of by the user as per the application requirements.

A PLLSLIP will cause a Missing Clock NMI if the REFCLKLO and REFCLKHI registers are configured exactly to catch the disturbances in the input frequency. However, if a PLLSLIP does not cause a Missing Clock NMI because the input to PLL is still within the range specified by the REFCLKLO and REFCLKHI limits, then the PLL will not be bypassed and ePWMs will not trip.

If a user wants to bypass the PLL on a PLLSLIP condition, it is suggested that the user configure the reference clock limits very strictly as shown in Table 1-15.

1.6.4 Control Subsystem PIE Vector Address Validity Check

To handle errors during an interrupt vector fetch, the features below are implemented on the PIE vector table memory:

- Two independent memories are reserved for the PIE vector table, one mapped at 0x0D00 and the other at 0x0E00 in C28 address space. Writes with address 0x0Dxx will get duplicated onto the memory at 0x0E00, whereas writes to the memory with address 0x0E00 will not get replicated to the memory at 0x0D00. This enables users to load the vector tables into both memories by just writing to memory at 0x0D00. Debug is possible during application testing since writes to the memory at 0xE00 are not duplicated in the memory at 0xD00. Dual memories are added mainly to detect more than one error, which is not possible with just parity. The following is the behavior of accesses to the PIE memories:
  - Data Writes at 0xD00: Writes to both memories
  - Data Writes at 0xE00: Writes only to the 0xE00 memory
  - Vector Fetch: Fetches from lower RAM at 0xD00 and compares with upper RAM at 0xE00
  - Data Read: Can read upper and lower RAM separately
- On every vector fetch from the PIE, a hardware comparison (no cycle penalty is incurred to do the comparison) of both vector table outputs is performed and if there is a mismatch between the two vector table outputs, the PIE returns a vector value that points to a fixed vector. This could be a vector location in C28 ROM or a fixed RAM or Flash location.
- Hardware also generates ePWM trip signals which trip the PWM outputs using TRIPIN15
- If there is no mismatch, the correct vector is jammed on to the C28x CPU program counter.
- In the event of mismatch, jamming a different vector address onto the C28x CPU program counter takes care of error handling in such a condition.

The following is the error handling expected to be done by the NMI error handler - configure standby mode in the C28 LPMCR0 register to enter stand by mode and execute IDLE instruction - C28 subsystem enters stand by mode.

In the event of a mismatch during a C28 NMI vector fetch, an NMI is generated to the M3 CPU.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: **[1]** The location which is jammed onto the C28x CPU program counter in case of normal vector fetch errors is 0x3FFFBE - this is just below the normal C28 vector table.

[2] Summarizing mismatch condition error handling:

- For NMI vector fetch:
  - Address 0x3FFFBE is jammed onto the CPU program counter.
  - NMI will be given to M3 system.
  - ePWM tripped
- For non-NMI vector fetch:
  - Address 0x3FFFBE is jammed on to the CPU program counter.
  - ePWM tripped

1.6.4.1 Boot ROM Handling of PIE Vector Address Validity Failure

The address 0x3FFFBE is located in the control subsystem's ROM (C-ROM), and a handler is installed by TI in C-Boot ROM. The handler installed by boot ROM at this location checks if this exception is happening because of a PIE VECTOR ADDRESS MISMATCH during NMI vector fetch or during a normal vector fetch. If it is due to an NMI vector fetch, the handler will clear the required NMI flags and send an IPC message to the master subsystem indicating the control subsystem is seeing a PIE vector address mismatch. A while (1) loop is entered without further executing any code on the control subsystem.

If this exception on the control subsystem is due to vector address mismatch while fetching a PIE interrupt vector, the master subsystem is not going to receive an interrupt. The user needs to depend on the IPC message sent by the control subsystem boot ROM. It is up to the user on how to handle this interrupt or IPC message from the control subsystem. The details on the IPC message sent by the control subsystem boot ROM installed handler are given in the *Boot ROM* guide chapter of this document.

If this exception on the control subsystem is due to a vector address mismatch while fetching an NMI interrupt vector, the master subsystem is going to get an NMI, and it is up to the user on how to handle this NMI. As mentioned earlier, the control subsystem boot ROM is going to clear the required NMI flags, so that it will not get reset by the CNMIWD. A while (1) loop is entered after sending an IPC to the master subsystem. In case the vector address mismatch is detected during an NMI fetch on the control subsystem, the master subsystem is going to get an NMI in addition to the IPC message from C-Boot ROM.

1.6.5 NMIWDs

Both the master subsystem and control subsystems have user-programmable NMIWD period registers in which users can set a limit on how much time they want to allocate for the device to acknowledge the NMI. If the NMI is not acknowledged, it will cause a device reset.

1.6.6 Watchdog Timers

The master subsystem has two watchdog timers, each operating on independent clocking domains (WDT0 uses M3SSCLK and WDT1 uses MAIN OSCCLK) with programmable interrupts and reset generation logic. The watchdog timer can be configured to generate an interrupt to the master subsystem on its first time-out, and 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. Please refer to the *Watchdog Timers* chapter for more details.

1.6.7 ECC and Parity Enabled RAMs, Shared RAMs Protection

RAM memories in both the master and control subsystems are ECC and parity enabled. All single bit errors in RAM are auto corrected and an error counter is incremented every time a single bit error is incremented. If the error counter reaches a predefined user configured limit then an interrupt is generated to each CPU. Refer to the *Internal Memory* chapter for more details on RAM errors.

Copyright © 2012–2019, Texas Instruments Incorporated

All uncorrectable double-bit errors end up triggering a BUS FAULT on the master subsystem's CPU and an NMI on the control subsystem. A RAMACCVIOL on the master subsystem triggers a BUSFAULT exception on and triggers a PIE interrupt on the control subsystem.

1.6.8 ECC Enabled Flash Memory

Flash single-bit errors are corrected automatically by ECC logic before giving data to the CPU, but they are not corrected in flash memory. Flash memory will still contain wrong data until another erase/program operation happens to correct the flash contents. Irrespective of whether the error interrupt is enabled or disabled, single bit errors are always corrected before giving data to the CPU. When the interrupt is disabled, users can check the single-bit error counter register (M3_ ERR_CNT) for any single bit error occurrences. The error counter stops incrementing once its value is equal to the threshold+1. It is always suggested to set the threshold register to a non-zero value so that the error counter can increment. It is up to the user to decide the threshold value at which they have to reprogram the flash with the correct data.

Flash uncorrectable errors end up triggering a BUSFAULT to the master subsystem's CPU and an NMI to the control subsystem. Please refer to the *Internal Memory* chapter of this document for more details on flash error correction and error catching mechanisms.

1.7 Power Control

The on-chip 1.2v and 1.8v VREGs generate the 1.2v VDD signal and 1.8v VDD signal from the 3.3 $_{VDDIO}$ input as needed to power the digital and analog subsystems on the device. This is done by pulling the -VREG12EN and -VREG18EN pins low. Refer to the device data manual for more details on these pins.

Users can also choose to provide an external 1.2v VDD signal and 1.8v VDD signal instead of using the on-chip VREGs. In this case, both the VREG12EN and VREG18EN pins should be pulled high.

The 3.3v input supply is monitored by the power-on rest (POR) circuit during a power-on condition. If a POR is detected, the $\overline{XRS}$ and $\overline{ARS}$ pins are pulled low to keep the device in reset.

The 1.8v VDD supply, when generated by the on-chip 1.8v VREG, is monitored by the power-on reset (POR) during a power-on condition. If a POR is detected, the $\overline{XRS}$ and $\overline{ARS}$ pins are pulled low to keep the device in reset.

The 1.2v VDD supply, when generated by the on-chip 1.2v VREG, is monitored by the power-on reset (POR) during a power-on condition. If a POR is detected, the $\overline{XRS}$ and $\overline{ARS}$ pins are pulled low to keep the device in reset.

1.8 Clock Control

The clocking control module has multiple sections covering the master subsystem clocking, control subsystem clocking, analog subsystem clocking, device low power modes clocking, and device attributes and configurations. There are configuration and status registers for each subsystems' clocking, device configurations, and low power mode configurations.

The master subsystem is responsible for clocking control and can read/write to clocking configuration registers by default on reset. The PLL and SYSDIVSEL registers are read only by default by the control subsystem, and the control subsystem can gain write-access to these registers by claiming the clock control semaphore register, CCLKREQUEST.

The master subsystem can also choose to switch off the clock for the control subsystem by setting the C28CLKINDIS (bit 0) of the CCLKOFF register. The clock to the control subsystem is enabled by default on power-up and after an $\overline{XRS}$.

Shared RAM, IPCs, and message RAMs, which are shared resources between the master and control subsystems, are clocked by the shared resource clock, which is the same as the PLLSYSCLK. Refer to the *Internal Memory* chapter for more details on shared resources.

Refer to the clocking diagrams in the master and control subsystems for more details on these clock sources.

Copyright © 2012–2019, Texas Instruments Incorporated

1.8.1 Clock Sources

There are four possible reference clock sources in this device:

- OSCCLK (from X1 or X1/X2 pins)
- GPIO_XCLKIN is a single ended clock input
- Internal 10 MHz OSCCLK (INTOSC)
- Internal 32 kHz clock derived from the internal 10 MHz oscillator

OSCCLK is always used as the reference clock source to generate the system clock. OSCCLK or GPIO_XCLKIN can be used to generate the 60 MHz USB clock. The INTOSC clock is used for missing clock condition detection and also to generate a low frequency 32 kHz clock in low power deep sleep modes. In deep sleep mode, the PLLs and X1/X2 are off.

The following sections give more details on each clock source.

1.8.1.1 OSCCLK

The oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the X1 input pin, or an external crystal is connected across the X1 input and X2 output pins. The crystal value must be one of the supported frequencies as mentioned in the device data manual. When used as a clock input to the system PLL, the single-ended clock source should be in the range mentioned in the device data manual. OSCCLK can also be a clock source for the USB PLL. A resonator can also be used as an input clock source on the X1/X2 pins.

1.8.1.2 GPIO_XCLKIN

GPIO_XCLKIN is a clock input which is available when the PJ7_GPIO63 pin is used as a clock input. This clock input can be used as a clock source only to the USB PLL and the CAN modules of the master subsystem.

1.8.1.3 10 MHz INTOSC

The internal oscillator is a 10 MHz on-chip clock source, which is used as the clock source in the event the main oscillator clock goes missing. It does not require the use of any external components and provides a clock that is 10 MHz. This clock should not be used as a normal run mode clock as it is just a clock for error handling in case the main OSCCLK goes missing. This clock can also be used as a clock source for the deep-sleep mode of the device if configured by the DSLPCLKCFG register.

1.8.1.4 32-kHz Clock

The 32-kHz clock is derived from the internal oscillator by clock division from 10 MHz to 32 kHz. This clock is intended for use during deep-sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the OSCCLK to be powered down if configured by the DSLPCLKCFG register.

The following limits apply to X1 and XCLKIN frequencies for DCAN0/1, WDT1, and USB peripherals:

- X1 can be clocked up to 100 MHz, if the clock input meets the expected duty cycle and jitter as mentioned in the data manual.
- XCLKIN input pin frequency is limited to 60 MHz.
- If DCAN0 or DCAN1 is clocked from the X1 (OSCLK) source, then it cannot exceed 100 MHz.
- If DCAN0 or DCAN1 is clocked from the XCLKIN source, then it cannot exceed 60 MHz.
- If the USB is clocked from the XCLKIN source, then it cannot exceed 60 MHz.

1.8.2 PLLs

There are two PLLs in this device; one used to generate the clocks for the entire device (system PLL) and the other to generate the fixed 60 MHz clock to the USB module (USB PLL). The clock input source to the system PLL is OSCCLK and USBPLL is OSCCLK or GPIO_XCLKIN. Both the PLLs are powered down in the deep sleep mode.

Copyright © 2012–2019, Texas Instruments Incorporated

The clocking control and low power mode control for the M3 and C28 subsystems are separate. This helps retain compatibility to C2000 devices on the C28 subsystem.

The PLL generates output frequency Fout, based on the following relationship with Fref, the input frequency. SYSPLLMULT also includes a fractional multiplication portion. Fout from the PLL is a 50% duty cycle clock, so there is no divider required at the output of the PLL in Concerto™ platform devices for normal operation.

Fout = Fref * SYSPLLMULT

The PLL by itself supports a Fout range of 110-550 MHz, but in Concerto ™devices, the maximum Fout is 300 MHz, and the maximum system clock (PLLSYSCLK) frequency supported is 150 MHz. There is a default /2 clock divider at the PLL output in addition to SYSDIVSEL and M3SSDIVSEL which generates PLLSYSCLK.

So,

PLLSYSCLK = Fref * SYSPLLMULT /2 /SYSDIVSEL divider

M3SSCLK = Fref * SYSPLLMULT / 2 / SYSDIVSEL divider / M3SYSDIVSEL divider

The PLL multiplier bit field is a 9 bit field with 7 bits of the SYSPLLMULT register comprising of the integer multiplier portion (SPLLIMULT) while the remaining 2 bits support the fractional multiplier (SPLLFMULT). The value of SPLLIMULT = 0 would default to PLLBYPASS mode where Fout = Fref. As an example,

if SPLLIMULT = 36 and SPLLFMULT = 2, it yields a multiplication factor of 36.5.

For a Fref = 4.11 MHz, the Fout = Fref * 36.5 = 150 MHz, which is further divided by /2, to generate a PLLSYSCLK frequency of 75 MHz.

NOTE:

1. The application must ensure that PLLCR values should be chosen such that Fout is always within a permissible Fout range, namely between 110 and 550 MHz.
2. The purpose of /2, /4 and /8 dividers (SYSDIVSEL) is to reduce inrush current at power up.

The SysCtlClockPllConfig () function in the sysctl.c file may be referenced as an example of a proper PLL initialization sequence. If this function is not used for PLL configuration, then it is recommended to lock the PLL twice whenever there is a change in the system clocking frequency, per the following sequence:

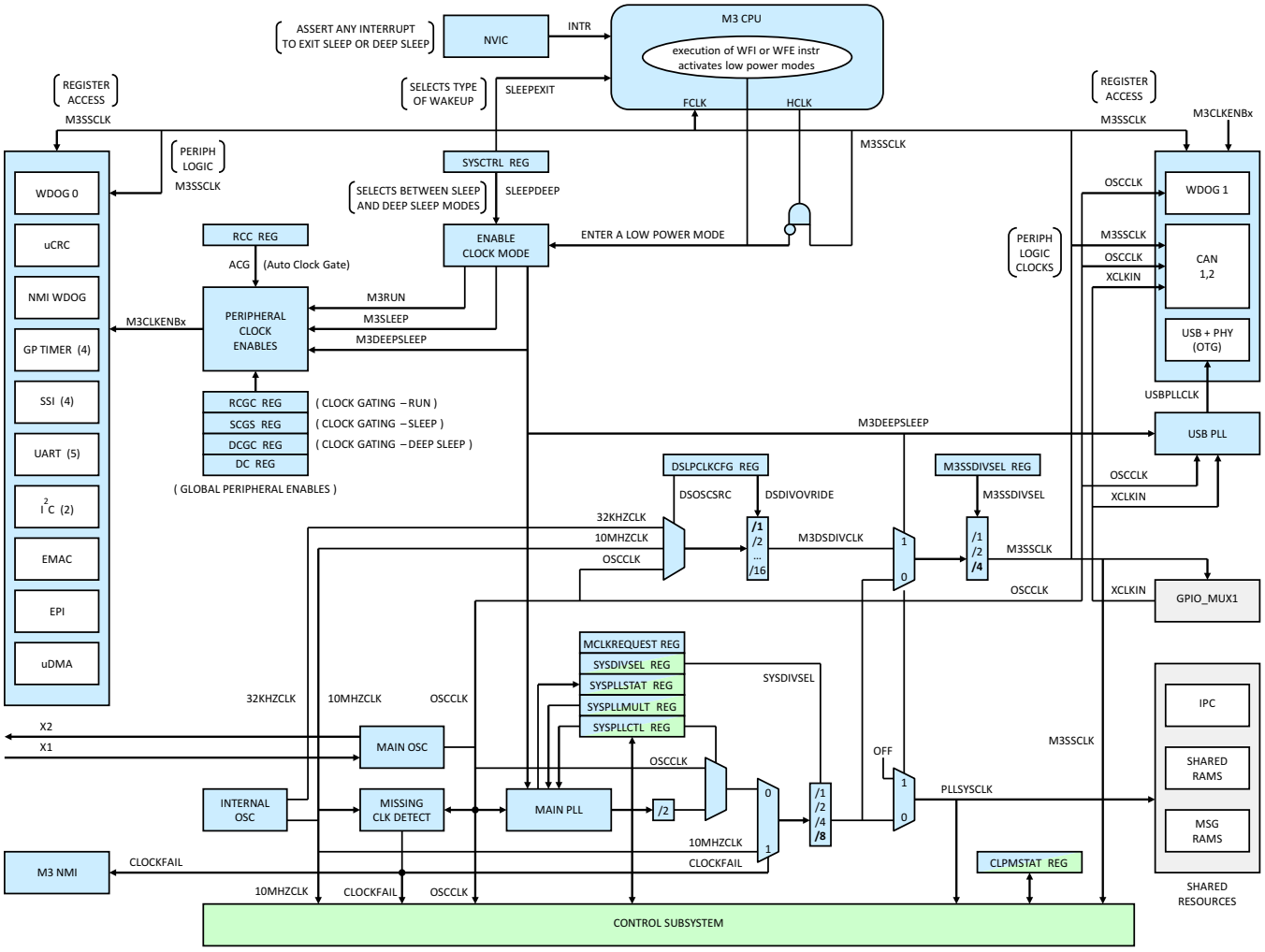
- Bypass the PLL
- Write the desired PLL multipliers to the SYSPLLMULT[IMULT] and SYSPLLMULT[FMULT] registers
- Poll the SYSPLLMULT[LOCKS] bit to wait for the first PLL lock sequence to complete
- Rewrite at least one of the multipliers (this will re-start the PLL Lock)
- Poll the lock bit again to ensure proper final PLL initialization
- Configure the system clock dividers
  - It is recommended to initially program a system divider which is greater than the final divide value (lower frequency) in the system. This will limit the instantaneous current increase when the PLL is selected as the clock source and allow the system voltage regulator to stabilize in smaller current step increments.
- Re-enable the PLL as the system clocking source
- Set the system dividers to the final values
  - The final divider can then be programmed after a small delay determined by the system regulator and decoupling capacitors stabilize

1.8.3 Master Subsystem Clocking

Figure 1-11 shows different clocks available in the master subsystem and explains how each clock is derived and the configuration registers involved.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-11. Master Subsystem Clocks and Low Power Mode Configuration



The internal PLLSYSCLK clock, normally used as a source for all clocks, is a divided-down output of the Main PLL (referred to as the System PLL in some cases) or X1 external clock input, as defined by the SPLLCKEN bit of the SYSPLLCTL register. There is also a second oscillator that internally generates two clocks: 32KHZCLK and 10MHZCLK, as shown in the figure above.

The 32KHZCLK, 10MHZCLK, and OSCCLK clocks are used by the master subsystem as possible sources for the deep sleep clock.

The Cortex-M3 master subsystem operates in one of three modes: run mode, sleep mode, or deep sleep mode. Refer to Section 1.9 for more details on sleep mode and deep sleep mode of operation.

As shown in Figure 1-11, PLLSYSCLK is either derived from the output of the PLL (when the PLL is enabled and locked) or from the MAIN OSC clock directly (when the PLL is bypassed or turned OFF) divided by the SYSDIVSEL divider. This PLLSYSCLK is the input clock for the control subsystem and the M3 system divider (M3SYSDIVSEL) whose output becomes M3SSCLK input to the master subsystem.

During Cortex-M3 normal mode of operation, the master subsystem is clocked by M3SSCLK and all the master subsystem peripherals when enabled are clocked by M3SSCLK as configured in the RCGCx registers. An exception is the Watchdog Timer 1 module which is clocked by OSCKCLK directly. The USBPLL and CAN modules have the option to choose a clock source other than OSCCLK as shown in Figure 1-11. Refer to the respective sections for more details on USB and CAN clocking configurations.

Copyright © 2012–2019, Texas Instruments Incorporated

For power saving purposes, the master subsystem Cortext-M3 RCGCx (run mode clock configuration), SCGCx (sleep mode clock configuration), and DSCGCx (deep sleep mode clock configuration) registers are used to control the clocks to each module, while the Cortext-M3 CPU is in run, sleep and deep sleep modes, respectively. These modes are defined below.

Run Mode	In run mode, the M3 CPU actively executes code. Run mode provides normal operation of the M3 CPU and all of the peripherals on the M3 subsystem, currently enabled by the RCGCx registers.
Sleep Mode	In sleep mode, the clock frequency of the active peripherals is unchanged, but the M3 CPU and its memory subsystem are not clocked and therefore no longer execute code. Sleep mode is entered by the Cortex-M3 core executing a WFI (Wait for Interrupt) instruction. Any properly configured interrupt event in the system brings the processor back into run mode. See the system control NVIC section of the *ARM® Cortex-M3 Technical Reference Manual* for more details. Peripherals enabled in the SCGCx register are clocked when auto-clock gating is enabled (RCC register) or the RCGCx register when the auto-clock gating is disabled. The M3 subsystem clock frequency is the same as during run mode. **Note:** This mode is equivalent to the conventional IDLE mode in C2000 devices.
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 M3 CPU clock being stopped. An interrupt returns the M3 CPU to run mode by a request from the code. Deep-sleep mode is entered by first writing the deep sleep enable bit in the ARM Cortex-M3 NVIC system control register and then executing a WFI instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See the system control NVIC section of the *ARM® Cortex-M3 Technical Reference Manual* for more details.

The device configuration registers which act as write masks to the above clock configuration registers are DC1, DC2, DC4, DC6, and DC10. Refer to Section 1.13.1 for more details on device configuration registers.

The Cortex-M3 processor and the memory subsystem are not clocked. Peripherals are clocked that are enabled in the DCGCx register when auto-clock gating is enabled (see the RCC register), or the RCGCx register when auto-clock gating is disabled. The system clock source in deep-sleep mode is specified in the DSLPCLKCFG register. The DSLPCLKCFG register is used to choose one of three clock sources, namely the MAIN OSCCLK (X1/X2) or the INTOSC (10 MHz) clock or 32 kHz clock. The main oscillator macros can be powered-down in deep sleep mode if either the 10 MHz or 32 kHz is chosen as the clock.

If the PLL is running at the time of the WFI instruction during deep-sleep mode, hardware powers the PLL down. The ACG bit in the RCC register determines the clock gating controls in deep sleep mode, and the clock frequency is determined by the DSDIVORIDE setting in the DSLPCLKCFG register with up to /16 or /64, respectively. When the deep-sleep exit event occurs, hardware brings the system clock back to the source and frequency it had at the onset of deep-sleep mode before enabling the clocks that had been stopped during the deep-sleep duration.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: **[1]** In deep sleep mode, if the application wants to wake up using the CAN or USB modules as configured by the respective DSCGC register bits, it is possible to do so. But the USB PLL is turned off, since generation of the wake up signal from the USB controller does not need a proper clock.

[2] The deep sleep clock as configured by the DSLPCLKCFG register, followed by the M3SSDIVSEL register, becomes the M3_SS_CLK input to the CANBCLKSEL logic. Software can decide to use either of the same three clock sources to the CAN bit clock as decided by the CANBCLKSEL register configuration.

[3] Below describes which clock will drive which peripherals in deep sleep mode:

- For M3 peripherals in deep sleep mode, the clock will be divided by the deep-sleep divider followed by the M3_SS divider.
- For C28 peripherals in deep sleep mode, the clock will be divided by the deep sleep divider only.

1.8.4 Control Subsystem Clocking

The input clock to the C28x processor is the C28CLKIN, (refer to Figure 1-12) which is the same as PLLSYSCLK coming from the master subsystem when clocking to the control subsystem is enabled (refer to the CCLKOFF register for more details). As mentioned previously, PLLSYSCLK is either derived from the output of the PLL (when the PLL is enabled and locked) or from the MAIN OSCCLK directly (when the PLL is bypassed or turned off), divided by the SYSDIVSEL divider.

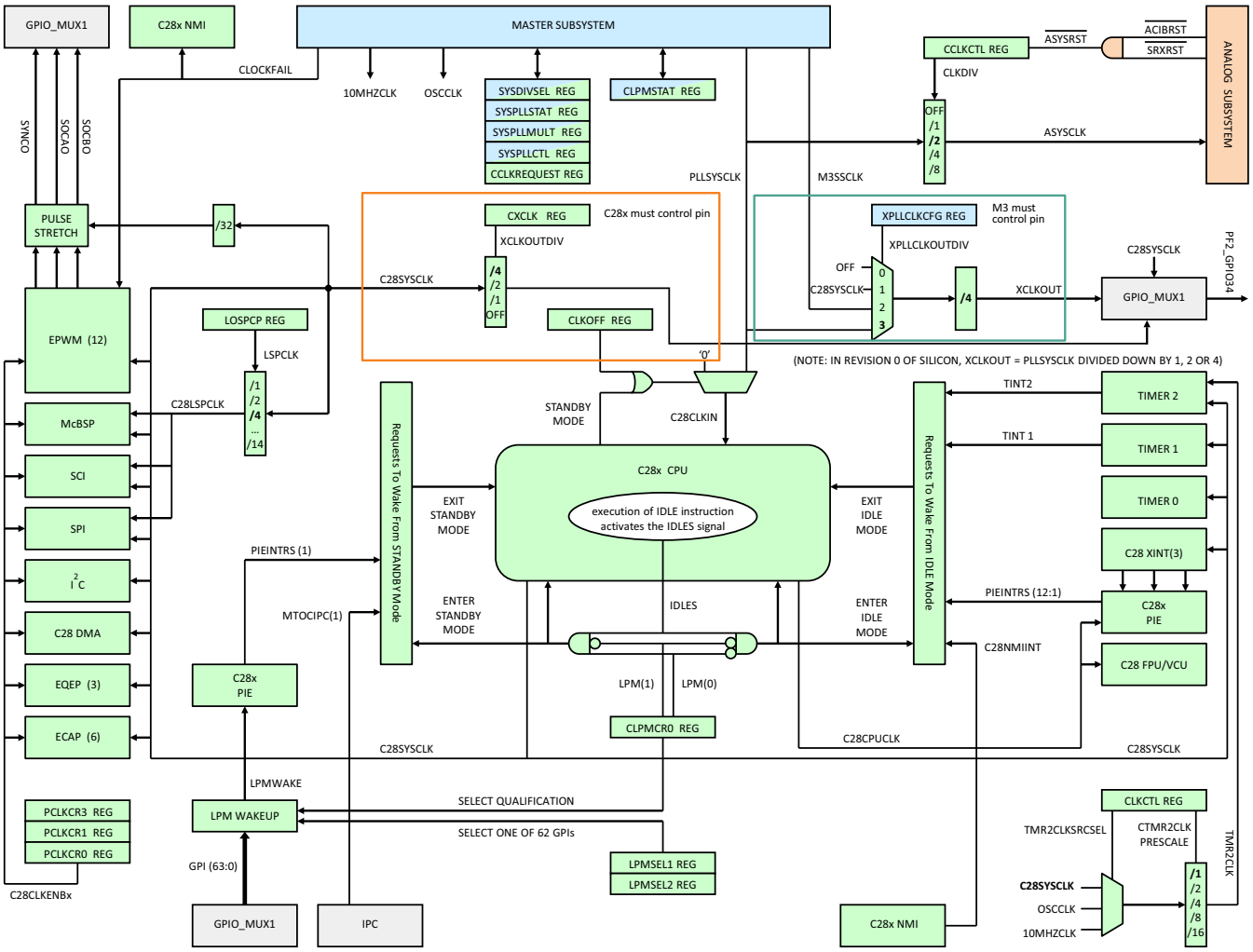
The control subsystem, by default after a reset, will not be able to configure the PLL and SYSDIVSEL registers. The PLL and SYSDIVSEL configuration is read-only for the control subsystem while the master subsystem can configure them as needed. To enable the ability to configure clocking from the control subsystem, there is a provision provided where the control subsystem can claim the ownership of clocking control and configure the PLL and SYSDIVSEL as needed.

The C28x processor outputs two clocks: C28CPUCLK and C28SYSCLK. The control subsystem operates in one of three modes: normal mode, idle mode, or standby mode.

Figure 1-12 shows the clocking control on the control subsystem.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-12. Control Subsystem Clocks and Low Power Mode Configuration



1.8.4.1 C28x Normal Mode

In normal mode, the C28x processor, most memory, and most of the peripherals are clocked by the C28SYSCLK, which is derived from the C28CLKIN input clock to the C28x processor. The remainder of the memories, FPU, VCU, PIE, and three timers are clocked by C28CPUCLK, which is also derived from the C28CLKIN. Timer 2 can be clocked by TMR2CLK, which is a divided-down version of one of three source clocks: C28SYSCLK, OSCCLK, or 10 MHZCLK as selected by the CLKCTL register.

Additionally, the LOSPCP register can be programmed to provide a low-speed clock (C28LSPCLK) to the SCI, SPI, and McBSP peripherals. Clock gating for individual peripherals is defined inside the PCLKCR0,1, and 3 registers. Execution of the IDLE instruction stops the C28x processor from clocking and activates the IDLES signal. The IDLES signal is gated with two LPM bits of the CLPMCR0 register to enter the control subsystem into idle mode or standby mode.

Refer to Section 1.9 for more details on these registers.

Copyright © 2012–2019, Texas Instruments Incorporated

1.8.4.2 C28x Idle Mode

In idle mode, the C28x processor stops executing instructions and the C28CPUCLK is turned off. The C28SYSCLK continues to run. Exit from idle mode is accomplished by any enabled interrupt or the C28NMIINT (C28x non-maskable interrupt). Upon exit from idle mode, the C28CPUCLK is restored. If LPMWAKE interrupt is enabled, the LPMWAKE ISR is executed. Next, the C28x processor starts fetching instructions from a location immediately following the IDLE instruction that originally triggered the idle mode.

1.8.4.3 C28x Standby Mode

In standby mode, the C28x processor stops executing instructions and the C28CLKIN, C28CPUCLK, and C28SYSCLK are turned off. Exit from Standby Mode is accomplished by one of 64 GPIOs from the GPIO_MUX block, or MTOCIPCINT2 (interrupt from the MTOC IPC peripheral). The wakeup GPIO selected inside the GPIO_MUX block enters the Qualification Block as the LPMWAKE signal. Inside the Qualification Block, the LPMWAKE signal is qualified per the QUALSTDBY bits (bits [7:2] of the CLPMCR0 register) before propagating into the wake request logic. Upon exit from STANDBY Mode, the C28CLKIN, C28SYSCLK, and C28CPUCLK are restored. If the LPMWAKE interrupt is enabled, the LPMWAKE ISR is executed. Next, the C28x processor starts fetching instructions from a location immediately following the IDLE instruction that originally triggered the Standby Mode.

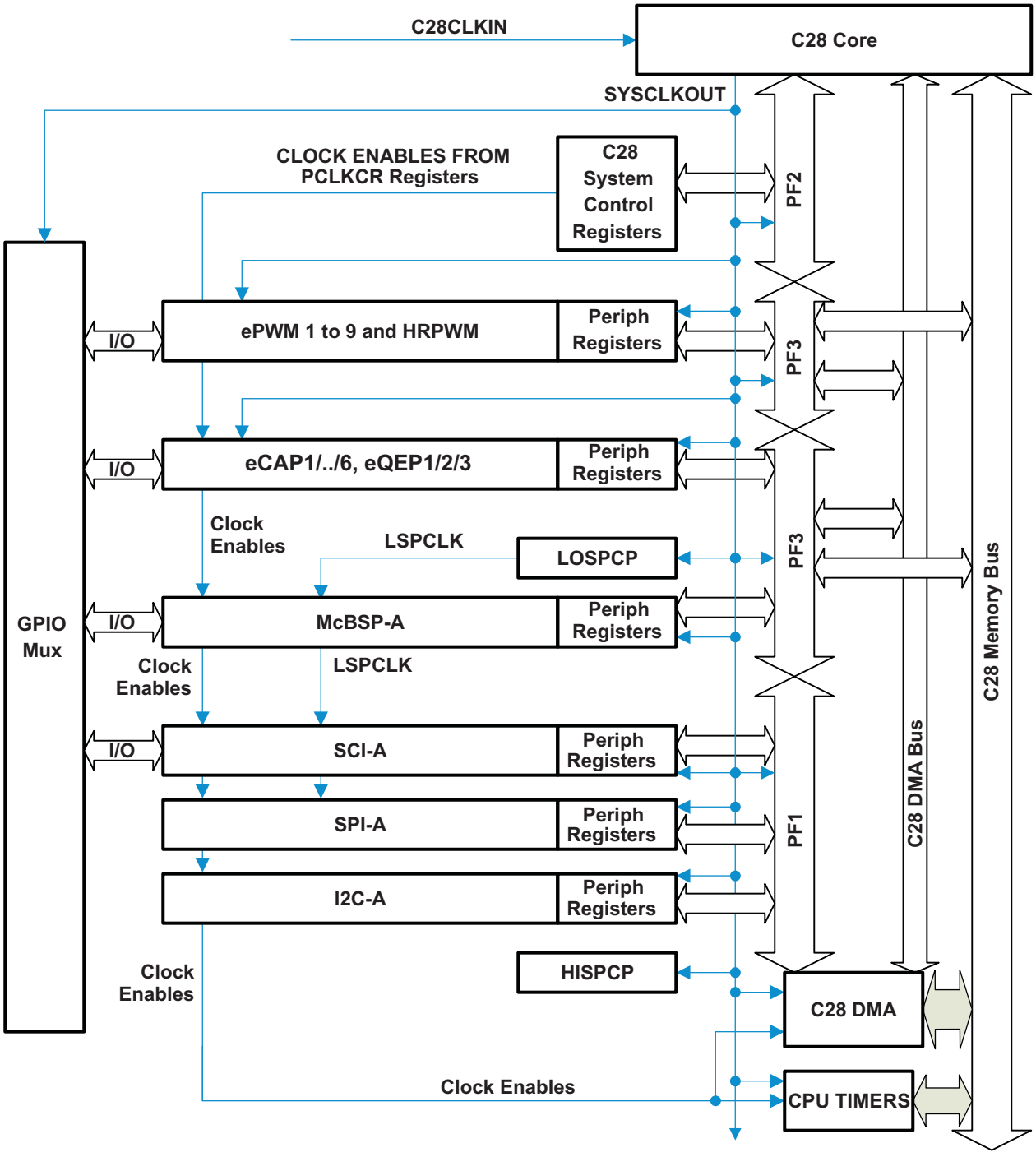
1.8.4.4 Control Peripherals Clocking

The clocking control for the C28 subsystem is done with the PCLKCR0, PCLKCR1, PCLKCR2, and PCLKCR3 registers. These registers are accessible only by the C28 CPU and are similar to earlier C2000 family of devices.

Figure 1-13 shows various clock domain on the control subsystem.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-13. Control Subsystem Peripherals Clocking



1.8.4.5 Enabling/Disabling Clocks to the Peripheral Modules

The PCLKCR0/1/2/3 registers enable/disable clocks to the various peripheral modules. There is a 2-SYSCLKOUT cycle delay from when a write to the PCLKCR0/1/2/3 registers occurs to when the action is valid. This delay must be taken into account before attempting to access the peripheral configuration registers. Due to the peripheral-GPIO multiplexing at the pin level, all peripherals cannot be used at the same time. While it is possible to turn on the clocks to all the peripherals at the same time, such a configuration may not be useful. If this is done, the current drawn will be more than required. To avoid this, only enable the clocks required by the application.

Copyright © 2012–2019, Texas Instruments Incorporated

1.8.4.6 Configuring the Low-Speed Peripheral Clock Prescaler

The low-speed peripheral clock prescale (CLOSPCP) registers are used to configure the low-speed peripheral clocks. See Figure 1-124 for the CLOSPCP layout.

1.8.4.7 Configuring Device Clock Domains

The CLKCTL register on the control subsystem can be used to choose between the available clock sources for the Timer 2 clock. TMR2CLK, as shown in Figure 1-13, is the clock source for Timer 2 on the control subsystem.

1.8.5 Clocking Control Semaphore Functionality

PLL configuration (SYSPLLCTL, SYSPLLMULT, SYSPLLSTS) and SYSDIVSEL configuration are only accessible by the master subsystem by default, but the control subsystem can request access to these registers which configure the PLL and SYSDIVSEL by using the clocking control semaphore logic. After the control subsystem is finished with clock configurations, it can return control back to the master subsystem. To request access to the control clock configuration and to return the control back to the previous owner of clock configurations, each subsystem is provided with a clock control semaphore register. This register is called MCLKREQUEST on the master subsystem, and CLKREQUEST on the control subsystem. Both registers always reflect the same value when both the master subsystem and control subsystem write to MCLKREQUEST and CCLKREQUEST, respectively, at the same time. Then, master subsystem writes are given priority.

By default on power-up or reset, the "SEM" bits (bits 1,0) in the MCLKREQUEST and CCLKREQUEST registers are set to 0,0. When these bits are "0,0" or "1,0" or "1,1" respectively, the master subsystem owns the clocking control semaphore and can read/write to the clocking control registers. When the master subsystem owns the clocking control semaphore, all the clocking control registers are READ-ONLY by the control subsystem.

When SEM bits are set to "0,1" respectively, the control subsystem owns the clocking control semaphore and can read/write to the clocking control registers. At this time, these registers are READ-ONLY for the master subsystem.

Registers which can be accessed by the subsystem which owns the clocking control semaphore currently are: SYSPLLCTL, SYSPLLMULT, SYSPLLSTS, and SYSDIVSEL.

For the master subsystem to own the clocking control registers:

- The SEM bits (bits 1,0) in the MCLKREQUEST register must be set to "0,0", "1,1" or "1,0" for the master subsystem to be able to read/write to the clocking control registers.
- If the SEM bits (bits 1,0) in the MCLKREQUEST register are set to "1,0", the clocking control semaphore is owned by the master subsystem. The control subsystem cannot gain this semaphore, and thus cannot gain access to the clocking control registers when the SEM bits are in this state.
- If the SEM bits (bits 1,0) in the MCLKREQUEST register are set to "0,1", the clocking control semaphore is owned by the control subsystem and the master subsystem cannot have access to the clocking control registers at this point. Therefore, it cannot set the SEM bits back to "0,0" or "1,1" or "1,0". Only the control subsystem can set the SEM bits to "0,0", "1,1" or "1,0" after they are set to "0,1."

If the user is configuring the clocking registers on the master subsystem, it is advised to set the SEM bits in the MCLKREQUEST register to "1,0" at least until clock setup is completed. This keeps the software running on the control subsystem from trying to gain access to the clocking registers while the master subsystem is configuring them.

For the control subsystem to own the clocking control registers:

- The SEM bits (bits 1,0) in the CCLKKREQUEST register must be set to "0,1" for the control subsystem to be able to read/write to the clocking control registers.
- Once the SEM bits are set to "0,1" in the CCLKREQUEST register by the control subsystem or in the MCLKREQUEST register by the master susbystem, then only the control subsystem will be able to set the SEM bits back to "0,0", "1,1" or "1,0" to give access back to the master subsystem.
- If the SEM bits (bits 1,0) in the CCLKREQUEST register are set to "1,0", the clocking control semaphore is with the master subsystem and the control subsystem cannot have access to the clocking control registers at this point. It also cannot set the SEM bits to "0,1". Only the master

Copyright © 2012–2019, Texas Instruments Incorporated

subsystem can set the SEM bits back to "0,0" or "1,1".

As explained above, it must be noted that the SEM bits in the CCLKREQUEST register can only be set to "0,1" when the current state of the SEM bits are either "0,0" or "1,1". Also the SEM bits in the MCLKREQUEST register can only be set to "1,0" when the current state of the SEM bits are either "0,0" or "1,1".

1.8.6 ACIB and Analog Peripherals Clocking

The ACIB and analog peripherals that are accessible through the ACIB are clocked by the PLLSYSCLK divided by the divider configured in the CCLKCTL register. As shown in Figure 1-12, this clock is referred to as ASYSCLK.

Only the control subsystem can configure the divider in the CCLKCTL register. Refer to the device data manual for more details on the ASYSCLK MIN/MAX requirements. To configure the clock for the analog subsystem, it is advised that the user call the TI-provided function in C-OTP (control subsystem OTP, refer to the memory map of this device). This function is explained in TI-provided analog OTP functions in the *Analog Subsystem* chapter. It is advised that this function is called with a proper divider value as a parameter based on user clock configuration requirements on the device during the application initialization process.

1.8.6.1 MCIBSTATUS and CCIBSTATUS

Both the master subsystem and control subsystem can monitor the status of the ACIB by looking at these registers. The CIBBUSCLKCNT bits (8-15) of both the MCIBSTATUS and CCIBSTATUS registers are incremented by the ASYSCLK. Users can use this counter to know if the ACIB is functional.

Bit 0 (APGOODSTS) of both these registers shows the status of the analog subsystem's power.

Bits 1 and 2 (READY and INTS bits) show the state of the ACIB READY and ACIB INTS signals which can be monitored for what caused an ACIBERR error.

The INTS signal will be stuck if there is no response on any of the analog peripheral interrupt lines that are connected through the ACIB. The READY signal will also be stuck if any of the reads/writes do not go through the ACIB.

1.8.7 Configuring XCLKOUT

To aid in debugging of the master subsystem and control subsystem, PF2_GPIO34 can be configured for XCLKOUT operation. Refer to the *GPIOs* chapter of this document on how to configure this GPIO for XCLKOUT operation.

XCLKOUT is PLLSYSCLK divided by the XPLLCLKOUTDIV set in the XPLLCLKCFG register as shown in Figure 1-12. Note that only the master subsystem can configure the XCLKOUT divider. The clock output on this GPIO pin is the output of PLLSYSCLK divided by the SYSDIVSEL divider.

Copyright © 2012–2019, Texas Instruments Incorporated

1.8.8 32-Bit CPU Timers 0/1/2

This section describes the three 32-bit CPU-timers (TIMER0/1/2) shown in (Figure 1-14).

CPU Timer-0 and CPU-Timer 1 can be used in user applications. Timer 2 is reserved for DSP/BIOS. If the application is not using DSP/BIOS, then Timer 2 can be used in the application. The CPU-timer interrupt signals (TINT0, TINT1, TINT2) are connected as shown in Figure 1-15.

Figure 1-14. CPU-Timers

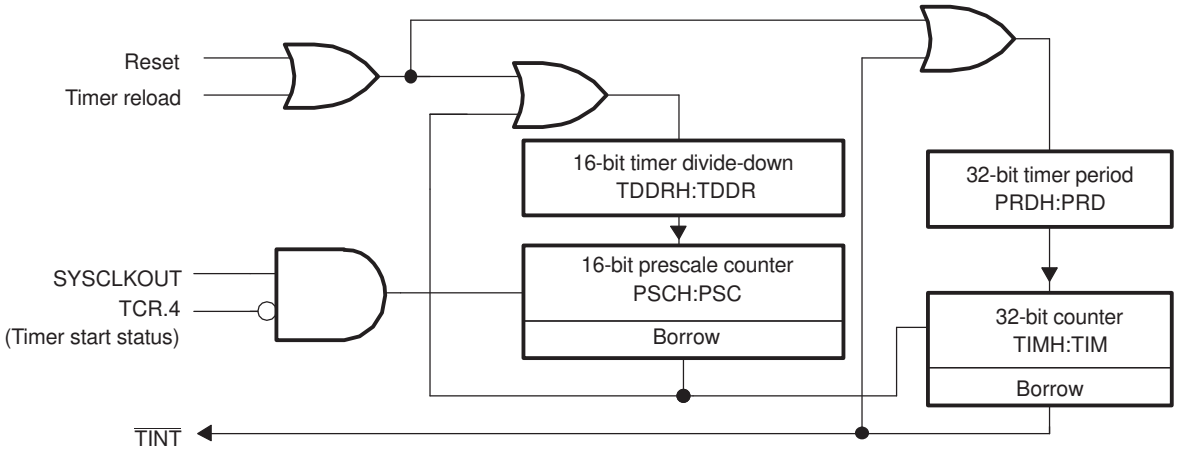
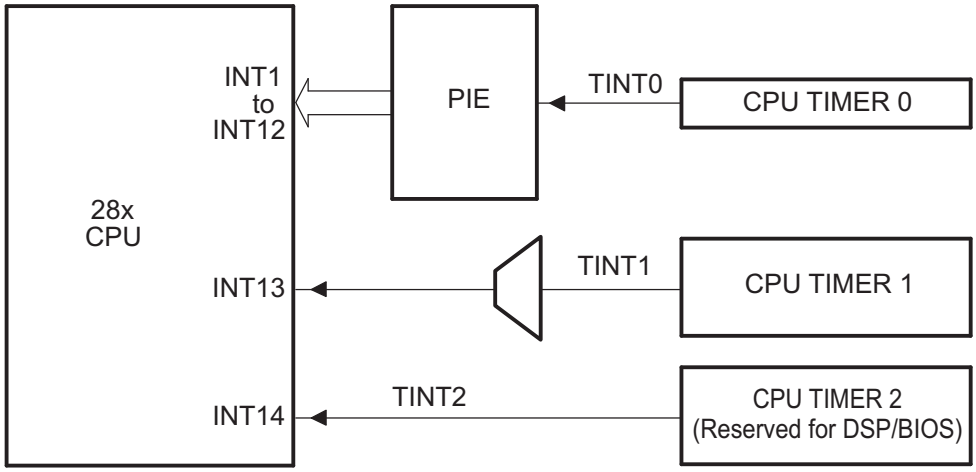


Figure 1-15. CPU-Timer Interrupts Signals and Output Signal



A The timer registers are connected to the Memory Bus of the C28x processor.

B The timing of the CPU timers are synchronized to SYSCLKOUT of the processor clock.

Copyright © 2012–2019, Texas Instruments Incorporated

The general operation of the CPU-timer is as follows: The 32-bit counter register TIMH:TIM is loaded with the value in the period register PRDH:PRD. The counter decrements once every (TPR[TDDRH:TDDR]+1) SYSCLKOUT cycles, where TDDRH:TDDR is the timer divider. When the counter reaches 0, a timer interrupt output signal generates an interrupt pulse. The registers listed in Table 1-16 are used to configure the timers.

Table 1-16. CPU-Timers 0, 1, 2 Configuration and Control Registers

Name	Address	Size (x16)	Description	Bit Description
TIMER0TIM	0x0C00	1	CPU-Timer 0, Counter Register	Figure 1-16
TIMER0TIMH	0x0C01	1	CPU-Timer 0, Counter Register High	Figure 1-17
TIMER0PRD	0x0C02	1	CPU-Timer 0, Period Register	Figure 1-18
TIMER0PRDH	0x0C03	1	CPU-Timer 0, Period Register High	Figure 1-19
TIMER0TCR	0x0C04	1	CPU-Timer 0, Control Register	Figure 1-20
TIMER0TPR	0x0C06	1	CPU-Timer 0, Prescale Register	Figure 1-21
TIMER0TPRH	0x0C07	1	CPU-Timer 0, Prescale Register High	Figure 1-22
TIMER1TIM	0x0C08	1	CPU-Timer 1, Counter Register	Figure 1-16
TIMER1TIMH	0x0C09	1	CPU-Timer 1, Counter Register High	Figure 1-17
TIMER1PRD	0x0C0A	1	CPU-Timer 1, Period Register	Figure 1-18
TIMER1PRDH	0x0C0B	1	CPU-Timer 1, Period Register High	Figure 1-19
TIMER1TCR	0x0C0C	1	CPU-Timer 1, Control Register	Figure 1-20
TIMER1TPR	0x0C0E	1	CPU-Timer 1, Prescale Register	Figure 1-21
TIMER1TPRH	0x0C0F	1	CPU-Timer 1, Prescale Register High	Figure 1-22
TIMER2TIM	0x0C10	1	CPU-Timer 2, Counter Register	Figure 1-16
TIMER2TIMH	0x0C11	1	CPU-Timer 2, Counter Register High	Figure 1-17
TIMER2PRD	0x0C12	1	CPU-Timer 2, Period Register	Figure 1-18
TIMER2PRDH	0x0C13	1	CPU-Timer 2, Period Register High	Figure 1-19
TIMER2TCR	0x0C14	1	CPU-Timer 2, Control Register	Figure 1-20
TIMER2TPR	0x0C16	1	CPU-Timer 2, Prescale Register	Figure 1-21
TIMER2TPRH	0x0C17	1	CPU-Timer 2, Prescale Register High	Figure 1-22

Figure 1-16. TIMERxTIM Register (x = 0, 1, 2)

15	0
TIM	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-17. TIMERxTIM Register Field Descriptions

Bits	Field	Description
15-0	TIM	CPU-Timer Counter Registers (TIMH:TIM): The TIM register holds the low 16 bits of the current 32-bit count of the timer. The TIMH register holds the high 16 bits of the current 32-bit count of the timer. The TIMH:TIM decrements by one every (TDDRH:TDDR+1) clock cycles, where TDDRH:TDDR is the timer prescale divide-down value. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers. The timer interrupt (TINT) signal is generated.

Figure 1-17. TIMERxTIMH Register (x = 0, 1, 2)

15	0
TIMH	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-18. TIMERxTIMH Register Field Descriptions

Bits	Field	Description
15-0	TIMH	See description for TIMERxTIM.

Figure 1-18. TIMERxPRD Register (x = 0, 1, 2)

15	0
PRD	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-19. TIMERxPRD Register Field Descriptions

Bits	Field	Description
15-0	PRD	CPU-Timer Period Registers (PRDH:PRD): The PRD register holds the low 16 bits of the 32-bit period. The PRDH register holds the high 16 bits of the 32-bit period. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers, at the start of the next timer input clock cycle (the output of the prescaler). The PRDH:PRD contents are also loaded into the TIMH:TIM when you set the timer reload bit (TRB) in the Timer Control Register (TCR).

Figure 1-19. TIMERxPRDH Register (x = 0, 1, 2)

15	0
PRDH	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-20. TIMERxPRDH Register Field Descriptions

Bits	Field	Description
15-0	PRDH	See description for TIMERxPRD

Figure 1-20. TIMERxTCR Register (x = 0, 1, 2)

15	14	13	12	11	10	9	8
TIF	TIE	Reserved		FREE	SOFT	Reserved	
R/W-0	R/W-0	R-0		R/W-0	R/W-0	R-0	

7	6	5	4	3	0
Reserved		TRB	TSS	Reserved	
R-0		R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-21. TIMERxTCR Register Field Descriptions

Bits	Field	Value	Description
15	TIF		CPU-Timer Interrupt Flag
			TIF indicates whether a timer overflow has happened since TIF was last cleared. TIF is not cleared automatically and does not need to be cleared to enable the next timer interrupt.
		0	The CPU-Timer has not decremented to zero.
			Writes of 0 are ignored.
		1	This flag gets set when the CPU-timer decrements to zero.
			Writing a 1 to this bit clears the flag.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-21. TIMERxTCR Register Field Descriptions (continued)

Bits	Field	Value	Description
14	TIE		CPU-Timer Interrupt Enable.
		0	The CPU-Timer interrupt is disabled.
		1	The CPU-Timer interrupt is enabled. If the timer decrements to zero, and TIE is set, the timer asserts its interrupt request.
13-12	Reserved		Reserved
11-10	FREE SOFT		CPU-Timer Emulation Modes: These bits are special emulation bits that determine the state of the timer when a breakpoint is encountered in the high-level language debugger. If the FREE bit is set to 1, then, upon a software breakpoint, the timer continues to run (that is, free runs). In this case, SOFT is a *don't care*. But if FREE is 0, then SOFT takes effect. In this case, if SOFT = 0, the timer halts the next time the TIMH:TIM decrements. If the SOFT bit is 1, then the timer halts when the TIMH:TIM has decremented to zero.
		FREE SOFT	CPU-Timer Emulation Mode
		0 0	Stop after the next decrement of the TIMH:TIM (hard stop)
		0 1	Stop after the TIMH:TIM decrements to 0 (soft stop)
		1 0	Free run
		1 1	Free run
			In the SOFT STOP mode, the timer generates an interrupt before shutting down (since reaching 0 is the interrupt causing condition).
9-6	Reserved		Reserved
5	TRB		CPU-Timer Reload bit.
		0	The TRB bit is always read as zero. Writes of 0 are ignored.
		1	When you write a 1 to TRB, the TIMH:TIM is loaded with the value in the PRDH:PRD, and the prescaler counter (PSCH:PSC) is loaded with the value in the timer divide-down register (TDDRH:TDDR).
4	TSS		CPU-Timer stop status bit. TSS is a 1-bit flag that stops or starts the CPU-timer.
		0	Reads of 0 indicate the CPU-timer is running.
			To start or restart the CPU-timer, set TSS to 0. At reset, TSS is cleared to 0 and the CPU-timer immediately starts.
		1	Reads of 1 indicate that the CPU-timer is stopped.
			To stop the CPU-timer, set TSS to 1.
3-0	Reserved		Reserved

Figure 1-21. TIMERxTPR Register (x = 0, 1, 2)

15	8	7	0
PSC		TDDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-22. TIMERxTPR Register Field Descriptions

Bits	Field	Description
15-8	PSC	CPU-Timer Prescale Counter. These bits hold the current prescale count for the timer. For every timer clock source cycle that the PSCH:PSC value is greater than 0, the PSCH:PSC decrements by one. One timer clock (output of the timer prescaler) cycle after the PSCH:PSC reaches 0, the PSCH:PSC is loaded with the contents of the TDDRH:TDDR, and the timer counter register (TIMH:TIM) decrements by one. The PSCH:PSC is also reloaded whenever the timer reload bit (TRB) is set by software. The PSCH:PSC can be checked by reading the register, but it cannot be set directly. It must get its value from the timer divide-down register (TDDRH:TDDR). At reset, the PSCH:PSC is set to 0.
7-0	TDDR	CPU-Timer Divide-Down. Every (TDDRH:TDDR + 1) timer clock source cycles, the timer counter register (TIMH:TIM) decrements by one. At reset, the TDDRH:TDDR bits are cleared to 0. To increase the overall timer count by an integer factor, write this factor minus one to the TDDRH:TDDR bits. When the prescaler counter (PSCH:PSC) value is 0, one timer clock source cycle later, the contents of the TDDRH:TDDR reload the PSCH:PSC, and the TIMH:TIM decrements by one. TDDRH:TDDR also reloads the PSCH:PSC whenever the timer reload bit (TRB) is set by software.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-22. TIMERxTPRH Register (x = 0, 1, 2)

15	8	7	0
PSCH		TDDRH	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-23. TIMERxTPRH Register Field Descriptions

Bits	Field	Description
15-8	PSCH	See description of TIMERxTPR.
7-0	TDDRH	See description of TIMERxTPR.

1.9 Low Power Modes

1.9.1 Low-Power Modes

The device can operate in various low-power modes. The entry and exit from low power modes on both the subsystems is controlled by separate logic, except in the deep sleep mode which is common to the entire device. All the below summarized low power modes are useful to conserve active power in the device.

Table 1-24 shows the state of the M3 and C28 subsystems for the low-power modes.

Table 1-24. Device Low Power Modes for Active Power Reduction

PLL	M3 CPU	Master Subsystem	C28 CPU	Control Subsystem
On (decided by SYSCLKCTL register) On	Sleep mode	Active (M3 SS CLK full speed)	Active (CLKIN, CPUCLK, SYSCLK on)	Active (SYSCLK on)
			Idle (CLKIN on, CPUCLK off, SYSCLK on)	Active (SYSCLK on)
			Standby (CLKIN off, CPUCLK off, SYSCLK off)	Stand by (SYSCLK off)
Off	Deep Sleep mode	Deep sleep mode	Standby (CLKIN off, CPUCLK off, SYSCLK off)	Stand by (SYSCLK off)

The master subsystem can be put in either sleep or deep-sleep low-power mode.

Allowed control subsystem modes when the master subsystem is in SLEEP mode are:

- Active mode
- Idle mode
- Standby mode

Allowed control subsystem mode when the master subsystem is in DEEPSLEEP mode is:

- Standby mode ONLY

It is expected that when the M3 subsystem is in the deep-sleep mode, the C28 subsystem is not in normal power mode. In other words, the M3 subsystem can enter deep sleep only when the C28 subsystem is in STANDBY mode. The low-power modes of individual subsystems are described in more detail in the following sections.

1.9.1.1 Master Subsystem Low-Power Modes Configuration

The master subsystem has the following low-power modes:

- Sleep mode
- Deep-sleep mode

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-25 summarizes the low-power modes on the M3 subsystem.

Table 1-25. M3 Subsystem Low-Power Modes

Mode	Enter Using	Oscillator clock	PLL (can be turned off / on depending on SYSCLKCTL register)	X1, X2	FCLK to M3 CPU	HCLK to M3 CPU	Exit
SLEEP(1)	Sleep Now WFE / WFI instructions	on	on	on	on	off	Any event for WFE / Any interrupt for WFI
	Sleep on Exit Set Sleep-on-exit bit of Cortex M3 SYSCTRL register +WFE/WFI	on	on	on	on	off	Any event for WFE / Any interrupt for WFI
DEEP SLEEP(1) (2)	Set SLEEPDEEP bit of Cortex M3 SYSCTRL register + Sleep now or Sleep-on-exit	On but lower frequency(3) (Refer to the DSLPCLKCFG register)	off	On	on	off	Any event for WFE / Any interrupt for WFI

(1) The low-power mode clock gating scheme will suppress the debug accesses in sleep and deep sleep mode.
(2) If the user exits deep-sleep mode, the user must not re-enter this mode within 5us otherwise the PLL timing may get affected.
(3) Turning OSC off in deep-sleep mode is not supported.

The SLEEPDEEP bit of the System Control (SYSCTRL) register selects which sleep mode is used. Refer to the SYSCTRL register description in the *Cortex-M3 Peripherals* chapter for more details.

1.9.1.1.1 Entering Low-Power Mode

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

The only distinguishing factor between selecting sleep or deep sleep as the system's low-power mode is the SLEEPDEEP bit of the M3 System Control (SYSCTRL) register. The value of this bit indicates the low-power mode selection for the system.

SLEEPDEEP Value	Description
0	Use sleep mode as the low power mode
1	Use deep-sleep mode as the low power mode

All the mechanisms below can be used to enter either sleep or deep sleep mode.

1.9.1.1.1.1 WAIT For INTERRUPT

The wait for interrupt instruction, WFI, causes immediate entry to sleep mode unless the wake-up condition is true (see Section 1.9.1.1.1.5). When the processor executes a WFI instruction, it stops executing instructions and enters sleep mode. See the *Cortex™-M3 Instruction Set Technical User's Manual* for more information.

Copyright © 2012–2019, Texas Instruments Incorporated

1.9.1.1.1.2 WAIT For EVENT

The wait for event instruction, WFE, causes entry to sleep mode depending on the value of a one-bit event register. When the processor executes a WFE instruction, it checks the event register. If the register is 0, the processor stops executing instructions and enters sleep mode. If the register is 1, the processor clears the register and continues executing instructions without entering sleep mode. If the event register is 1, the processor must not enter sleep mode on execution of a WFE instruction. Typically, this situation occurs if an SEV instruction has been executed. Software cannot access this register directly. See the Cortex™-M3 Instruction Set Technical User's Manual for more information.

1.9.1.1.1.3 SLEEP-ON-EXIT

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

1.9.1.1.1.4 WAKE UP From Low-Power Mode

The conditions for the processor to wake up depend on the mechanism that cause it to enter low-power mode.

1.9.1.1.1.5 WAKE UP From WFI or SLEEP-ON-EXIT

Normally, the processor wakes up only when it detects an exception with sufficient priority to cause exception entry. Some embedded systems might have to execute system restore tasks after the processor wakes up and before executing an interrupt handler. Entry to the interrupt handler can be delayed by setting the PRIMASK bit and clearing the FAULTMASK bit. If an interrupt arrives that is enabled and has a higher priority than a current exception priority, the processor wakes up but does not execute the interrupt handler until the processor clears the PRIMASK bit. For more information about the PRIMASK and FAULTMASK registers, refer to the register descriptions section under the *Cortex-M3 Processor*, Programming Model section.

1.9.1.1.1.6 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, Please refer to the register description under the System Control Block section of the Cortex-M3 Peripherals chapter.

NOTE: When the M3 subsystem and the device are in deep-sleep mode, wake-up should only happen from the M3 subsystem since it is the master. Wake-up should not happen from the C28 subsystem. This needs to be taken care of by the application.

The clocking is explained in detail in the Clock Control section. For more information about the clocking of the sleep modes, see the Master Subsystem Clocking and Control Subsystem Clocking sections. For more details on M3 power modes, please refer to the *Power Management Chapter of the Cortex™-M3 Technical Reference Manual*.

1.9.1.2 Control Subsystem Low-Power Modes Configuration

Table 1-26 summarizes the various low-power modes.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-26. Low-Power Modes Configuration

Mode	C28 LPMCR0 register Bits [1:0]	OSCCLK	CLKIN to C28 CPU	CPUCLK	SYSCLKOUT	Exit [1]
Normal	X,X	On	On	On	On	-
IDLE	0,0	On	On	Off	On	XRSn[2], Any Enabled C28 Interrupt,C28 NMI
STANDBY[3]]	0,1	On	Off	Off	Off	XRSn[2] ,GPIO0.async to GPIO63.async[4] MTOCIPCINT2

(1) The Exit column lists which signals or under what conditions the low power mode will be exited. A low signal, on any of the signals, will exit the low power condition. This signal must be kept low long enough for an interrupt to be recognized by the device. Otherwise the IDLE mode will not be exited and the device will go back into the indicated low power mode

(2) If the control subsystem receives an $\overline{\text{XRS}}$ while in low power mode, it will reset the control subsystem.

(3) In the device, the M3 CPU is the master. It is expected that when the M3 subsystem is in deep-sleep mode, the C28 subsystem will also be in a low power mode. When the M3 subsystem is in deep-sleep mode, the C28 subsystem is not expected to be in normal power mode. In other words, the M3 subsystem can enter deep-sleep only when the C28 subsystem is in STANDBY mode.

(4) The GPIO0.async to GPIO63.async are the asynchronous version of the input signal which come straight from the pin before any qualification or synchronization inside GPIO. The qualification is done with the OSC clock inside the C28 LPM block

1.9.1.2.1 Entering Low-Power Mode

The system enters low-power mode upon execution of the IDLE instruction. The low-power mode bits (LPM, [1:0]) of the CLPMCR0 register are only valid when the IDLE instruction is executed. The user must set the LPM bits to the appropriate mode before executing the IDLE instruction.

Refer to the *TMS320C28x DSP CPU and Instruction Set Reference Guide*, SPRU430, which describes the assembly language instructions of the TMS320C28x device.

1.9.1.2.1.1 IDLE Mode

If the CLPMCR0 bits [1:0] are set to (0, 0) and upon execution of an IDLE instruction, the processor stops executing further and enters into IDLE mode. The LPM block performs no tasks during this mode until a valid wakeup condition occurs.

1.9.1.2.1.2 Standby Mode

If the CLPMCR0 bits [1:0] are set to (0, 1) and upon execution of an IDLE instruction, the processor stops executing further and enters into STANDBY mode. The LPM block performs no tasks during this mode until a valid wakeup condition occurs. Refer to the *Low-Power Modes Registers* section of this document for CLPMCR0 register details.

1.9.1.2.2 Wake UP From Low-Power Mode

1.9.1.2.2.1 Idle Mode

This mode is exited by any enabled C28 interrupt that is recognized by the processor or the C28NMIINT (C28x Non-maskable interrupt).

1.9.1.2.2.2 Standby Mode

Any GPIO port signal (0 to 63) or MTOCIPCINT2 (interrupt from MTOC IPC peripheral) can wake the device from STANDBY mode. To use the GPIO wakeup, the user must select which signal(s) will wake the device in the GPIOLPMSELx register. The selected signal(s) are also qualified by the OSCCLK before waking the control subsystem, depending on the QUALSTDBY bits (bits [7:2] of the CLPMCR0 register) .

Refer to the *GPIOs* chapter for more details on the GPIOLPMSELx registers.

Copyright © 2012–2019, Texas Instruments Incorporated

1.10 Code Security Module (CSM)

The Code Security Module (CSM) is a security feature incorporated in this device. It prevents direct data accesses of on-chip secure memories (and other secure resources) by unauthorized entities. Unsecure or unlocked resources will still remain accessible by any means; for example, through a debugging tool such as Code Composer Studio™(CCS).

DISCLAIMER: CODE SECURITY MODULE DISCLAIMER

The Code Security Module (CSM) included on this device was designed to help protect the contents stored in the associated memory (password protection).

TI DOES NOT, HOWEVER, WARRANT OR REPRESENT THAT THE CSM CANNOT BE COMPROMISED OR BREACHED OR THAT THE CONTENTS STORED IN THE ASSOCIATED MEMORY CANNOT BE ACCESSED THROUGH OTHER MEANS. THE CSM HELPS TO ADDRESS CERTAIN SECURITY THREATS TO CODE CONFIDENTIALITY, BUT THE CUSTOMER IS SOLELY RESPONSIBLE FOR THE SECURITY OF THEIR DATA AND SYSTEMS AND NEEDS TO IMPLEMENT SUFFICIENT SECURITY MEASURES.

1.10.1 Functional Description

The security module restricts CPU access to on-chip secure memory without interrupting or stalling CPU execution. When a read occurs to a protected memory location, the read returns a zero value and CPU execution continues with the next instruction. This, in effect, blocks read and write access to various memories through the JTAG port or external peripherals. Security is defined with respect to the access of on-chip secure memories and prevents unauthorized copying of proprietary code or data.

The memory zone is secure when CPU access to the on-chip secure memories associated with that zone is restricted. When secure, two levels of protection are possible depending on where the program counter is currently pointing. If code is currently running from inside secure memory, only an access through JTAG is blocked (that is, through the debug probe). This allows secure code to access secure data. Conversely, if code is running from unsecure memory, all accesses to secure memories are blocked. User code can dynamically jump in and out of secure memory, thereby allowing secure function calls from unsecure memory. Similarly, interrupt service routines can be placed in secure memory, even if the main program loop is run from unsecure memory.

The code security mechanism present in this device offers dual zone security for the Cortex-M3 code and single zone security for the C28x code. In case of dual zone security on the master subsystem, different secure memories (RAMs and flash sectors) can be assigned to different security zones by configuring the GRABRAM and GRABSECT registers assocated with each zone. Flash sector N and flash sector A are dedicated to zone1 and zone2, respectively, and cannot be allocated to any other zone by configuration registers. Table 1-27 shows the status of a RAM based on the configuration in the GRABRAM register.

Similarly, flash sectors get assigned to different zones based on the setting in the GRABSECT registers.

Table 1-27. Master Subsystem Secure RAM Zone Selection

GRAB-Cn bits in Z1_GRABRAMR Register	GRAB-Cn bits in Z2_GRABRAMR Register	Ownership
00	XX	Cx RAM is inaccessible
XX	00	Cx RAM is inaccessible
Differential Value(01/10)	Differential Value (01/10)	Cx RAM is inaccessible
Differential Value(01/10)	11	Cx RAM belongs to zone1
11	Differential Value(01/10)	Cx RAM belongs to zone2
11	11	Cx RAM is non-secure

Security is provided by a CSM password of 128-bits (four 32-bit words) that is used to secure or unsecure the zones. Each zone has its own 128 bit CSM password. The zone can be unsecured by executing the password match flow (PMF). Table 1-28 shows the levels of security.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-28. Security Levels

PMF executed with correct password?	Operating Mode of the Zone	Program Fetch Location	Security Description
No	Secure	Outside secure memory	Only instruction fetches by the CPU are allowed to secure memory. In other words, code can still be executed, but not read.
No	Secure	Inside secure memory	CPU has full access. The JTAG port cannot read the secured memory contents.
Yes	Non-Secure	Anywhere	Full access for the CPU and JTAG port to secure memory of that zone.

The CSM password for each zone is stored in their respective dedicated flash sectors. These locations store the CSM password pre-determined by the system designer.

If the password locations of a zone have all 128 bits as ones, the zone is unsecure. Since new flash devices have erased flash (all ones), only a read of the password locations is required to bring any zone into unsecure mode. If the password locations of a zone have all 128 bits as zeros, the zone is secure, regardless of the contents of the CSMKEY registers. The user should not use all zeros as a password or reset the device during an erase of the flash. Resetting the device during an erase routine can result in either an all zero or unknown password. If a device is reset when the password locations are all zeros, the device cannot be unlocked by the password match flow described in Section 1.10.3.2. Using a password of all zeros will seriously limit user's ability to debug secure code or re-program the flash.

NOTE: If a device is reset while the password locations of a zone are all zeros or an unknown value, that zone will be permanently locked unless a method to run the flash erase routine from secure SARAM is embedded into the flash or OTP. Care must be taken when implementing this procedure to avoid introducing a security hole.

1.10.1.1 Emulation Code Security Logic (ECSL)

In addition to the CSM, the emulation code security logic (ECSL) has been implemented using a 64-bit password for each zone to prevent unauthorized users from stepping through secure code. Any code or data access to on-chip secure memories while the debug probe is connected will trip the ECSL and break the emulation connection. Like the CSM password, these passwords are also stored in flash memory and the value of password is predetermined by the system designer. To allow emulation of secure code, while maintaining the CSM protection against secure memory reads, the user must write the correct password into the ECSLKEY registers, which matches the password value stored in the flash. This will disable the ECSL. Also, if the value at the ECSL password locations in flash are all ones (unprogrammed), then ECSL gets disabled after ECSL password locations are read, irrespective of the values in ECSLKEY registers. Unlocking of the CSM also unlocks ECSL irrespective of the values in the ECSLKEY registers.

When initially debugging a device with the password locations in flash programmed (that is, secured), the debug probe takes some time to take control of the CPU. During this time, the CPU will start running and may execute an instruction that performs an access to a protected ECSL area. If this happens, the ECSL will trip and cause the debug probe to be disconnected.

Two solutions to this problem exist:

1. The first is to use the Wait-In-Reset emulation mode, which will hold the device in reset until the debug probe takes control. The debug probe must support this mode for this option.
2. The second option is to use the "branch to check boot mode" boot option. In this mode, the core will be in a loop and continuously poll the boot mode select pins. You can select this boot mode and then exit this mode once the debug probe is connected by re-mapping the PC to another address or by changing the boot mode selection pin to the desired boot mode.

NOTE: Access to the secure memory from the debuuger does not trip the debug probe. These accesses are just blocked and return '0'.

Copyright © 2012–2019, Texas Instruments Incorporated

1.10.1.2 Execute-Only Protection

To achieve a higher level of security on flash sectors which store critical user code (instruction opcodes), an execute-only protection feature is provided on this device. When the execute-only protection is turned on for any flash sector, data reads to that flash sector are disallowed from any code (even from secure code). Execute-only protection for a flash sector can be turned on by programming the corresponding EXEONLY-SECT bit to 1 in the Zx_EXEONLY location in flash memory. A dummy read of the Zx_EXEONLY location loads the bit fields associated with that particular sector in the zones (which has ownership of that sector) EXEONLYR register.

NOTE: Use of the execute-only security mode with the Coretx-M3 introduces some complications. When the Cortex-M3 C code is compiled and linked, literal data (constants, and so on) are 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 decode 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, inhibits correct execution. To insure correct execution in this case, the user must ensure that literal data is always placed into one or more read-enabled flash blocks.

1.10.1.3 JTAG Lock

The JTAG lock feature on the device can be used to disable the JTAG accesses (debugger accesses) permanently on the device. The user can enable the JTAG lock feature by programming the OTP_JTAGLOCK field with any value other than “1111” (0xF) at the OTPSEC location in OTP. This feature takes effect only after the OTPSEC location in OTP is read. Though the JTAG connection is not blocked before the OTPSEC location is read, access (all types) to all memories on the device are disabled until security is initialized (see the steps listed in Section 1.10.2).

CAUTION

If the JTAG lock feature is enabled, all future debugs of the device through JTAG will be disabled. This will specifically impair TI’s ability to analyze devices returned to TI for failure analysis. If this feature is enabled, TI may reject any return analysis requests.

1.10.1.4 Password Lock

Each zone’s password locations (CSM & ECSL) can be locked by programming the zone’s PSWDLOCK field with any value other than “1111” (0xF) at the OTPSEC location in OTP. Untill passwords of a zone are locked, password locations will not be secure and will have full access. This means that the debugger as well as code running from non-secure memory, can read the password locations. This feature can be used by the user to avoid accidental locking of zones while programming the flash sectors during the software development phase. On a fresh device, the value for password lock fields for all zones at OTPSECLOCK locations in OTP will be “1111”, which means passwords for all zones will be unlocked.

NOTE: The PSWDLOCK value only affects the password locations such that, when unlocked it only makes the password locations unsecure. All other secure memories and the other locations of the flash sectors which contain the password remain secure, as per the security settings. But since the passwords are unsecure, anyone can read it and unsecure the zone by running through the PMF. Hence, a user must program the PSWDLOCK location to change it from the default value of “1111”. Along with programming the PSWDLOCK field, the user should also enable the EXEONLY feature on flash sector A, which contains all of the security settings, for complete security initialization.

Copyright © 2012–2019, Texas Instruments Incorporated

1.10.2 CSM Impact on Other On-Chip Resources

On this device, not all memories on the master subsystem and control subsystem are secure (for example, C2/C3 on the master system and M0/M1/L2/L3 on the control subsystem and Shared RAM). To avoid any potential hacking when the device is in a default state (post reset), accesses (all types) to all memories (secure as well as non-secure, except Boot ROM & OTP) and to the analog subsystem, are disabled until proper security initialization is done. That is, just after reset, none of the memory resources except boot ROM and OTP are accessible.

The following are the steps required after reset (any type of reset) to initialize security on the device.

- Dummy Read to address location OTPSECLOCK in OTP
- Dummy read to address location Z1_GRABSECT in Flash
- Dummy read to address location Z1_GRABRAM in Flash
- Dummy read to address location Z1_EXEONLY in Flash
- Dummy read to address location Z2_GRABSECT in Flash
- Dummy read to address location Z2_GRABRAM in Flash
- Dummy read to address location Z2_EXEONLY in Flash

Control subsystem

- Dummy Read to address location OTPSECLOCK in OTP by M3 software
- Dummy read to address location EXEONLY in flash

1.10.3 Incorporating Code Security in User Applications

Code security is typically not employed in the development phase of a project; however, security may be desired once the application code is finalized. Before such code is programmed in the flash memory, a CSM password should be chosen to secure the zone. Once a CSM password is in place for a zone, the zone is secured (that is, programming a password at the appropriate locations and either performing a device reset or setting the FORCESEC bit (CSMSCR.15) is the action that secures the device). From that time on, access to debug the contents of secure memory by any means (via JTAG, code running off external/on-chip memory, and so on) requires a valid password. A password is not needed to run the code out of secure memory (such as in end-application usage); however, access to secure memory contents for debug purpose requires a password.

If the code-security feature is used, any one of the following directives must be used when a function residing in secure memory calls another function which belongs to a different secure zone or to unsecure memory:

- Use unsecure memory as stack
- Switch stack to unsecure memory before calling the function
- Unlock security before calling the function

Note that the above directives apply for any address-based-parameters passed on to the called function, basically making sure that the called function can read/write to these address-based parameters.

Table 1-29. OTPSECLOCK - Reserved Locations in OTP Memory

Memory Address	Register Name	Reset Values	Register Description
0x68-0800	OTPSECLOCK	User Defined	Contains one time programmable settings for security.

Table 1-30. M3 Zone1 - Reserved Locations in Flash Memory

Memory Address	Register Name	Reset Values	Register Description
0x20-0000	Z1_CSMPSWD0	User Defined	Low word (32-bit) of the 128-bit CSM password for zone1

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-30. M3 Zone1 - Reserved Locations in Flash Memory (continued)

Memory Address	Register Name	Reset Values	Register Description
0x20-0004	Z1_CSMPSWD1	User Defined	Second word (32-bit) of the 128-bit CSM password for zone1
0x20-0008	Z1_CSMPSWD2	User Defined	Third word (32-bit) of the 128-bit CSM password forzZone1
0x20-000C	Z1_CSMPSWD3	User Defined	High word (32-bit) of the 128-bit CSM password for zone1
0x20-0010	Z1_ECSLPSWD0	User Defined	Low word (32-bit) of the 64-bit CSM password for zone1
0x20-0014	Z1_ECSLPSWD1	User Defined	High word (32-bit) of the 64-bit CSM password for zone1
0x20-0018	Z1_GRABSECT	User Defined	Zone select settings for flash sectors for zone1
0x20-001C	Z1_GRABRAM	User Defined	Zone select settings for secure RAM for zone1
0x20-0020	Z1_EXEONLY	User Defined	Exe-only settings for flash sectors for zone1

Table 1-31. M3 Zone2 - Reserved Locations in Flash Memory

Memory Address	Register Name	Reset Values	Register Description
0x2F-FFF0	Z2_CSMPSWD0	User Defined	Low word (32-bit) of the 128-bit CSM password for zone2
0x27-FFF4	Z2_CSMPSWD1	User Defined	Second word (32-bit) of the 128-bit CSM password for zone2
0x27-FFF8	Z2_CSMPSWD2	User Defined	Third word (32-bit) of the 128-bit CSM password for zone2
0x27-FFFC	Z2_CSMPSWD3	User Defined	High word (32-bit) of the 128-bit CSM password for zone2
0x27-FFE8	Z2_ECSLPSWD0	User Defined	Low word (32-bit) of the 64-bit CSM password for zone2
0x27-FFEC	Z2_ECSLPSWD1	User Defined	High word (32-bit) of the 64-bit CSM password for zone2
0x27-FFE4	Z2_GRABSECT	User Defined	Zone select settings for flash sectors for zone2
0x27-FFE0	Z2_GRABRAM	User Defined	Zone select settings for secure RAM for zone2
0x27-FFDC	Z2_EXEONLY	User Defined	Exe-only settings for flash sectors for zone2

Table 1-32. C28x - Reserved Locations in Flash Memory

Memory Address	Register Name	Reset Values	Register Description
0x13-FFF8	CSMPSWD0	User Defined	Low word (32-bit) of the 128-bit CSM password
0x13-FFFA	CSMPSWD1	User Defined	Second word (32-bit) of the 128-bit CSM password
0x13-FFFC	CSMPSWD2	User Defined	Third word (32-bit) of the 128-bit CSM password
0x13-FFFE	CSMPSWD3	User Defined	High word (32-bit) of the 128-bit CSM password
0x13-FFF4	ECSLPSWD0	User Defined	Low word (32-bit) of the 64-bit CSM password

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-32. C28x - Reserved Locations in Flash Memory (continued)

Memory Address	Register Name	Reset Values	Register Description
0x13-FFF6	ECSLPSWD1	User Defined	High word (32-bit) of the 64-bit CSM password
0x13-FFF2	EXEONLY	User Defined	Exe-Only settings for flash sectors

1.10.3.1 Environments That Require Security Unlocking

The following are the typical situations under which unsecuring memories can be required:

- Code development using debuggers (such as Code Composer Studio™). This is the most common environment during the design phase of a product.
- Flash programming using TI's flash utilities such as UniFlash or Code Composer Studio™ F28xx On-Chip Flash Programmer plug-in.

Flash programming is common during code development and testing. Once the user supplies the necessary password, the flash utilities disable the security logic before attempting to program the flash. The flash utilities can disable the code security logic in new devices without any authorization, since new devices come with an erased flash. However, reprogramming devices (that already contain a custom password) require the password to be supplied to the flash utilities in order to unlock the device to enable programming. In custom programming solutions that use the flash API supplied by TI, unlocking the CSM can be avoided by executing the flash programming algorithms from secure memory.

- Custom environments defined by the application, in addition to the above, access to secure memory contents can be required in situations such as:
  - Using the on-chip bootloader to load code or data into secure SARAM or to erase/program the flash.
  - Executing code from on-chip unsecure memory and requiring access to secure memory for lookup table. This is not a suggested operating condition as supplying the password from external code could compromise code security.

The unsecuring sequence is identical in all the above situations. This sequence is referred to as the password match flow (PMF) for simplicity. Figure 1-23 explains the sequence of operation that is required each time the user attempts to unsecure a particular zone. A code example is listed for clarity.

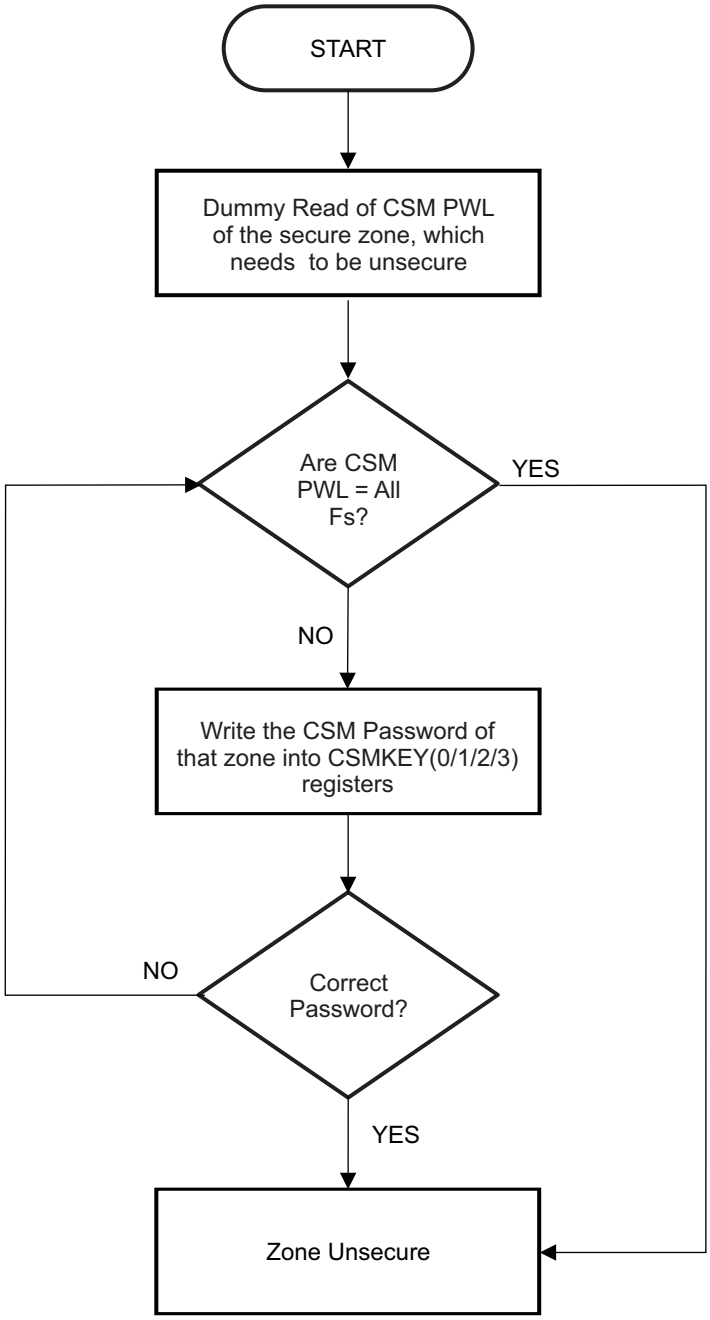
NOTE: If you want to erase/program secure flash sectors without unlocking the zone, the erase/program code must run from secure RAM linked to same zone as of secure flash sectors. In case of master subsystem (which has dual-zone), you also need to program the SEM bit in SECZONEREQUEST register. Please refer to SECZONEREQUEST register section for details about the SEM bit.

1.10.3.2 CSM Password Match Flow

Password match flow (PMF) is essentially a sequence of eight 16-bit (or four 32-bit) dummy reads from password locations (PWL) followed by eight 16-bit (or four 32-bit) writes to CSMKEY registers. Figure 1-23 shows how PMF helps to initialize the security logic registers and disable security logic.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-23. CSM Password Match Flow



Copyright © 2012–2019, Texas Instruments Incorporated

1.10.3.3 Unsecuring Considerations for Zones With/Without Code Security

Case 1 and Case 2 provide unsecuring considerations for zones with and without code security.

Case 1: Zone With Code Security

A zone with code security should have a predetermined password stored in the password locations of that zone. The following are steps to unsecure any secure zone:

1. Perform a dummy read of the password locations of that zone.
2. Write the password into the CSMKEY registers.
3. If the password is correct, the zone becomes unsecure; otherwise, it stays secure.

Case 2: Zone Without Code Security

A zone without code security should have 0x FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF (128 bits of all ones) stored in the password locations. The following are steps to use this zone:

1. At reset, the CSM will lock memory regions protected by the CSM.
2. Perform a dummy read of the password locations.
3. Since the password is all ones, this alone will unlock the zone, and all secure memories dedicated to that zone are fully accessible immediately after this operation is completed.

NOTE: Even if a zone is not protected with a password (all password locations all ones), the CSM will lock at reset. Thus, a dummy read operation must still be performed on these zones prior to reading, writing, or programming secure memory if the code performing the access is executing from outside of the CSM protected memory region. The Boot ROM code does this dummy read for convenience.

1.10.3.3.1 C Code Example to Unsecure C28x Zone1

```
volatile long int *CSM = (volatile long int *)0x000AE0; //CSM register file
volatile long int *CSMPWL = (volatile long int *)0x0013FFF8; //CSM Password location
volatile int tmp;
int I;
// Read the 128-bits of the CSM password locations (PWL)
//
for (i=0; i<4; i++) tmp = *CSMPWL++;
// If the password locations (CSMPWL) are all = ones (0xFFFF),
// then the zone will now be unsecure. If the password
// is not all ones (0xFFFF), then the code below is required
// to unsecure the CSM.
// Write the 128-bit password to the CSMKEY registers
// If this password matches that stored in the
// CSLPWL then the CSM will become unsecure. If it does not
// match, then the zone will remain secure.
// An example password of:
// 0x11112222333344445555666677778888 is used.
*CSM++ = 0x22221111; // Register CSMKEY0 at 0xAE0
*CSM++ = 0x44443333; // Register CSMKEY1 at 0xAE2
*CSM++ = 0x66665555; // Register CSMKEY2 at 0xAE4
*CSM++ = 0x88887777; // Register CSMKEY3 at 0xAE6
```


Copyright © 2012–2019, Texas Instruments Incorporated

1.10.3.3.2 C Code Example to Resecure C28x Zone1

```
volatile int *CSMSCR = 0x00AEF; //CSMSCR register
//Set FORCESEC bit
*CSMSCR = 0x8000;
```

Table 1-33 shows different conditions for a zone to be secure or non-secure.

Table 1-33. Zone Security Status

CSM-ARMED	CSM-ALLZERO	CSM-ALLONE	CSM-MATCH	ZONE STATUS
0	X	X	X	SECURE
1	0	0	0	SECURE
1	1	0	X	SECURE
1	0	1	X	NON-SECURE
1	0	0	1	NON-SECURE

1.10.3.4 Environments That Require ECSL Unlocking

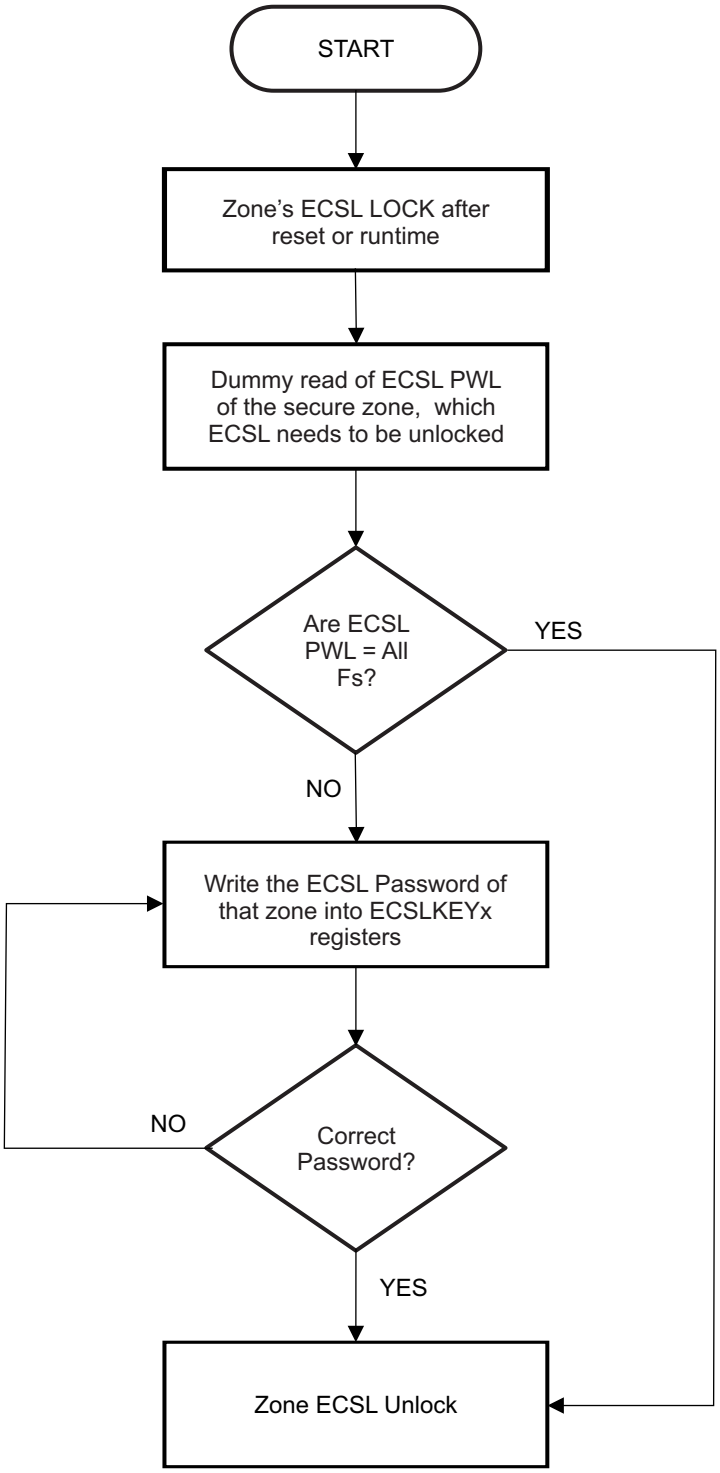
The following are the typical situations under which unsecuring memories can be required: the user develops some main IP, and then outsources peripheral functions to a subcontractor who must be able to run the customer code during debug but source code should not be visible to the subcontractor.

1.10.3.5 ECSL Password Match Flow

Password match flow (PMF) is essentially a sequence of eight dummy reads from password locations (PWL) followed by eight writes to KEY registers. Figure 1-24 shows how the PMF helps to initialize the security logic registers and disable security logic.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-24. ECSL Password Match Flow



Copyright © 2012–2019, Texas Instruments Incorporated

1.10.3.6 ECSL Disable Considerations for any Zone

Case 1 and Case 2 provide ECSL disable considerations for any Zone.

Case 1:

A secure zone with ECSL enabled should have a predetermined ECSL password stored in the ECSL password locations in flash. The following are steps to disable the ECSL for any particular zone:

1. Perform a dummy read of ECSL password locations of that zone.
2. Write the password into the ECSLKEYx registers corresponding to that zone.
3. If the password is correct, the ECSL gets disabled; otherwise, it stays enabled.

Case 2:

A secure zone with ECSL disabled should have 0x FFFF FFFF FFFF FFFF (64 bits of all ones) stored in the ECSL password locations of that zone. The following are steps to use this zone:

1. At reset, the ECSL will be locked hence any access to secure memory will terminate the JTAG connection.
2. Perform a dummy read of the ECSL password locations.
3. Since the ECSL password is all ones, this alone will unlock the specific secure zone. Now onwards any access to secure memories associated with that particular zone will not cause termination of JTAG connection.

NOTE: Even if a secure zone's ECSL is not locked with a password (all password locations all ones), the ECSL will be locked at reset. Thus, a dummy read operation must still be performed for that zone prior to any access to the memories associated with that zone. The Boot ROM code does these dummy reads for all the zones for convenience. Also if the CSM of a zone gets unlocked, ECSL also gets disabled for that zone.

1.10.3.6.1 C Code Example to Disable ECSL for C28x-Zone1

```
volatile long int *ECSL = (volatile int *)0x000AF0; //ECSL register file
volatile long int *ECSLPWL = (volatile int *)0x0013FFF4; //ECSL Password location
volatile int tmp;
int I;
// Read the 64-bits of the password locations (PWL)
// in flash at address 0x3F 7FF8 - 0x3F 7FFF
// If the zone is secure, then the values read will
// not actually be loaded into the temp variable, so
// this is called a dummy read.
for (i=0; i<2; i++) tmp = *ECSLPWL++;
// If the ECSL password locations (ECSLPWL) are all = ones (0xFFFF),
// then the ECSL will now be disbale. If the password
// is not all ones (0xFFFF), then the code below is required
// to disable the ECSL.
// Write the 64-bit password to the ECSLKEY registers
// If this password matches that stored in the
// ECSLPWL then ECSL will get disable. If it does not
// match, then the zone will remain secure.
// An example password of:
// 0x1111222233334444 is used.
*ECSL++ = 0x22221111; // Register ECSLKEY0 at 0xAF0
*ECSL++ = 0x44443333; // Register ECSLKEY1 at 0xAF2
```

Table 1-34 shows different conditions for any zone's ECSL to be secure or non-secure.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-34. Zone ECSL Status

ECSL-ARMED	ECSL-ALLZERO	ECSL-ALLONE	ECSL-MATCH	ZONE ECSL
0	X	X	X	ENABLE
1	0	0	0	ENABLE
1	1	0	X	ENABLE
1	0	1	X	DISBALE
1	0	0	1	DISBALE

1.10.4 Do's and Don'ts to Protect Security Logic

1.10.4.1 Do's

- Recheck the password stored in the password locations before programming the COFF file using flash utilities.
- Program the PASSWDLOCK field of the OTPSECLOCK register with a value other than “0x1111” to secure the device. The flow of code execution can freely toggle back and forth between secure memory and unsecure memory without compromising security.

1.10.4.2 Don'ts

- If code security is desired, do not embed the password in your application anywhere other than in the password locations or security can be compromised.
- Do not use 128 bits of all zeros as the password. This automatically secures the zone regardless of the contents of the CSMKEY register. The code in that zone is not debuggable nor reprogrammable.
- Do not pull a reset during an erase operation on the flash array. This can leave either zeros or an unknown value in the password locations. If the password locations are all zero during a reset, the zone will always be secure regardless of the contents of the KEY register

1.11 µCRC Module

The µCRC module is part of the master subsystem. This module can be used by M3 software to compute CRC on data/program data, stored at memory locations which are addressible by the M3 subsystem. The M3 flash bank and ROM are mapped to the code space which is only accessed by the ICODE/DCODE bus of the M3. RAMs are mapped on the SRAM space which is accessible by the SYSTEM bus. Hence, the µCRC module snoops both the DCODE and SYSTEM buses to support CRC calculation for data/program data.

1.11.1 Functional Description

The µCRC module snoops both the DCODE and SYSTEM buses to support CRC calculation for data/program data. To allow interrupts to execute in between CRC calculations for a block of data, and to discard the Cortex-M3 literal pool accesses in between execution of the program, (which reads data for CRC calculation) the M3 ROM, Flash, and RAMs are mapped to a mirrored memory location (refer to the device data manual for addresses of mirrored memory space). The µCRC module grabs data from the bus to calculate CRC only if the address of read data belongs to mirrored memory space. After grabbing the data, the µCRC module performs the CRC calculation on grabbed data and updates the µCRCRES register. This register can be read any time to get the calculated CRC for all the previous read data at any time. The µCRC module only supports CRC calculation for byte accesses. This means to calculate the CRC on a block of data, software must do byte accesses to all the data to calculate CRC. For half-word and word accesses, the µCRC module discards the data and does not update.

NOTE: If a read to mirrored address space is thrown from the debugger (CCS or any other debug platform), the uCRC module ignores the read data and does not update the CRC result for that perticual read.

Copyright © 2012–2019, Texas Instruments Incorporated

1.11.2 CRC Polynomials

The following are the CRC polynomials supported by the µCRC module:

CRC8 Polynomial = 0x07

CRC16 Polynomial-1 = 0x8005

CRC16 Polynomial-2 = 0x1021

CRC32 Polynomial = 0x04C11DB7

1.11.3 CRC Calculation Procedure

The following is the software procedure to calculate CRC for a set of data stored in M3 addressable memory space:

- Save the current value of the CRC result register (uCRCRES) into the stack to allow calculation of CRC in nested interrupt.
- Clear the CRC result register (uCRCRES) by setting the CLEAR field of the uCRCCONTROL register to '1'.
- Configure uCRC polynomials (CRC8, CRC16-P1, CRC16-P2 or CRC32) in the uCRCCONFIG register.
- Read the data from memory locations for which CRC needs to be calculated using mirrored address.
- Read the µCRCRES register to get the calculated CRC value. Pop the last save value of the CRC from stack and store it into the CRC result register (uCRCRES).

1.11.4 CRC Calculation For Data Stored In Secure Memory

This device has dual zone security for the M3 subsystem (refer to Section 1.10 for more details). Since zoneX (X -> 1/2) software does not have access to data/program data in zoneY (Y -> 2/1), code running from zoneX cannot calculate CRC on data stored in zoneY memory.

Similarly, in case of Exe-Only flash sectors, even though software is running from same secure zone, it cannot read the data stored in Exe-Only sectors. Hardware does allow CRC computation on data stored in Exe-Only flash sectors as long as the read access for this data is initiated by code running from same secure zone. These reads are just dummy reads, and in this case, read data only goes to the µCRC module, not to the CPU.

1.12 Inter Processor Communications (IPC)

The Inter Processor Communications (IPC) module facilitates communication between the master and control subsystems. This section details the IPC mechanisms that the master (M3) subsystem and control (C28x) subsystem can use to request/share information between the two subsystems and notify the status of any dependent tasks between the two subsystems.

This module consists of six main features:

- MSGRAMs
- IPC Flags and Interrupts
- IPC Message Registers
- Flash Pump Semaphore
- Clock Configuration Semaphore
- Free running counter

Along with the above features, the IPC module has some boot registers that are used by boot ROM code for identifying boot modes and boot statuses. Refer to the *Boot ROM* chapter for more information on these IPC boot registers.

Copyright © 2012–2019, Texas Instruments Incorporated

1.12.1 MSGRAMs

There is a dedicated 2KB MTOC message RAM (MTOCMSGRAM), from which the M3 can read/write and the C28x can read. There is another dedicated 2KB CTOM-message RAM (CTOMMSGRAM) from which the C28x can read/write and the M3 can read. For MTOCMSGRAM, the M3 CPU and µDMA have read and write accesses, whereas the C28 CPU and DMA have only read access. Similarly for CTOMMSGRAM, the C28 CPU and DMA will have read and write accesses, whereas the M3 CPU and µDMA will have only read access. See Table 1-35.

Table 1-35. IPC MSG RAM Read/Write Accesses

	C28x	M3	28x DMA	M3 µDMA
C28x to M3 Message RAM (2KB)	R/W	R	R/W	R
M3 to C28x Message RAM (2KB)	R	R/W	R	R/W

For safety, the master subsystem DMA's (µDMA in the master subsystem is the master for the MTOCMSGRAM and the DMA in the control subsystem is the master for CTOMMSGRAM) write access to a MSGRAM is made configurable using the MTOCMSGRCR and CTOMMSGRCR registers for the M3 and C28x cores, respectively. See the *Internal Memory* chapter for more information on these registers.

There is no hardware relationship between the IPC flags/interrupts and the message RAMs. Users can use them together to accomplish better communication techniques between the cores/subsystems. A protocol to use the message RAMs with the IPC flags/interrupts needs to be established in software. For more details on MSGRAM initialization and configuration, refer to the *Internal Memory* chapter.

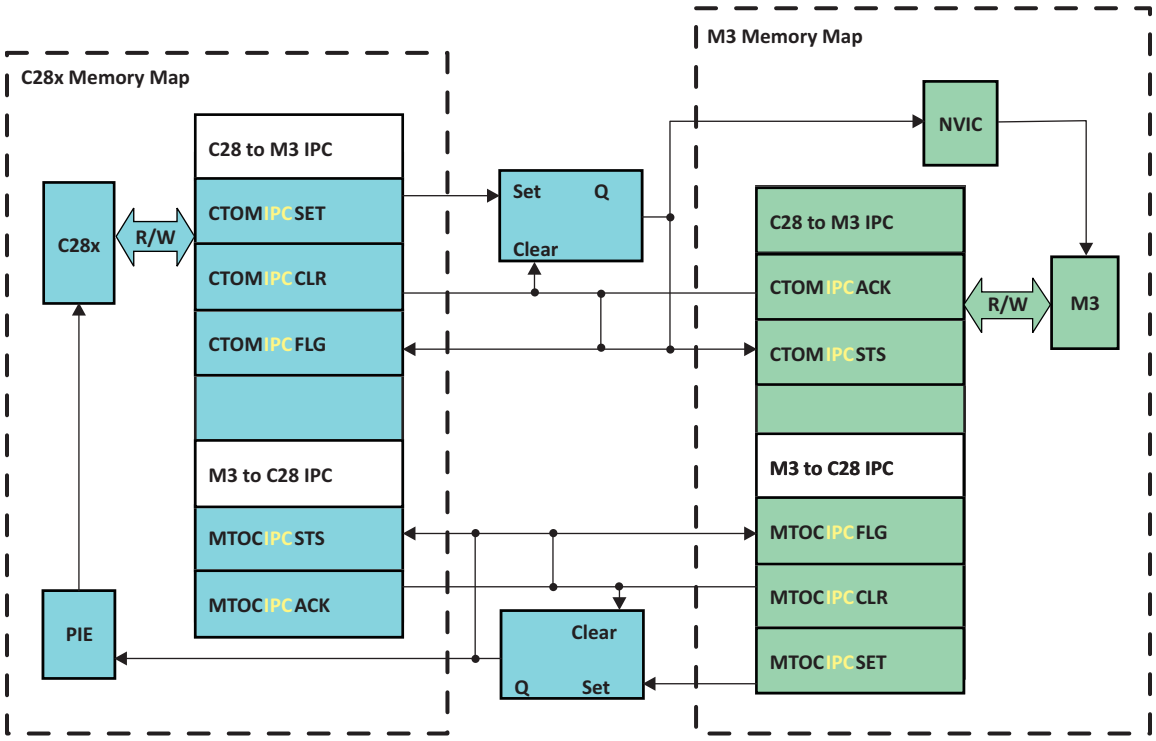
These message RAMs can be used to pass messages between the two subsystems. When the master subsystem writes a message in MTOCMSGRAM, the M3 can use either MTOCIPC flags or interrupts to indicate to the control subsystem that the message is ready for the C28x to read. Similarly, when the control subsystem writes a message in CTOMMSGRAM, the C28x can use either CTOMIPC flags or interrupts to indicate to the master subsystem that the message is ready for the M3 to read.

1.12.2 IPC Flags and Interrupts

There are 32 IPC handshake channels from the master system to the control system and vice versa to enable communication between the cores based on software flags. Out of these 32, four channels (1 through 4) can be enabled to generate IPC interrupts to the other core. These handshake channels can be used along with the Message RAMs to build a software handshake mechanism between the two cores. Figure 1-25 shows the IPC flag messaging and interrupt mechanism.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-25. Messaging with IPC Flags and Interrupts



1.12.3 MTOCIPC Communication

When a message is ready for the M3 to send to the C28x, the M3 can notify that to the C28x using MTOCIPC flags or interrupts (flags 1-4). The C28x, upon polling for these MTOCIPC flags or upon getting an MTOCIPC interrupt, can respond to the message sent by the M3. There are three 32-bit registers, MTOCIPCSET, MTOCIPCCLR, and MTOCIPCFLG on the M3 memory map and two 32-bit registers, MTOCIPCSTS and MTOCIPCACK, to implement the flag-based message communication from the M3 to the C28x. The MTOCIPCSET register has 32 bits, each of which when set, is capable of enabling a corresponding flag bit in the MTOCIPCFLG register on the M3 and a status bit in the MTOCIPCSTS register on the C28x.

For example:

1. When the M3 wants to notify a message readiness to the C28x, the M3 can set bit 9 (IPC flag 10) in the MTOCIPCSET register.
2. This action of the M3 will result in setting bit 9 in the MTOCIPCSTS register on the C28x along with bit 9 in the MTOCIPCFLG register on the M3. The MTOCIPCFLG register on the M3 and the MTOCIPCSTS register on the C28x are physically the same, but named differently as the FLG register on the M3 and the STS register on the C28x.
3. The C28x can poll for bit 9 (IPC flag 10) of the MTOCIPCSTS register while waiting for the message. When bit 9 of the MTOCIPCSTS register gets set, the C28x can read the message if any or can take necessary action depending on the design of the user's application.
4. Then the C28x should acknowledge the M3 that it received the MTOCIPC request by setting bit 9 of the MTOCIPCACK register which will result in clearing bit 9 in the MTOCIPCFLG register and MTOCIPCSTS register.

   Note that the MTOCIPCCLR register on the M3 and the MTOCIPCACK register on the C28x are physically the same, but named differently as the CLR register on the M3 and the ACK register on the C28x.

Copyright © 2012–2019, Texas Instruments Incorporated

For every bit in the MTOCIPCSET register, there is a bit in the MTOCIPCCLR register. The M3 can set bits in the MTOCIPCCLR register to clear corresponding flags (flag bit in MTOCIPCFLG and status bit in MTOCIPCSTS) that the M3 has set previously using the MTOCIPCSET register. Generally, the C28x should acknowledge the MTOCIPC request using the MTOCIPCACK register which will clear the MTOCIPCFLG bits and MTOCIPCSTS bits. However, at times when the M3 did not receive any acknowledgement from the C28x about the reception of a message in an expected duration, the M3 application software might want to clear the MTOCIPC flag request that it set earlier. In such a scenario, the M3 can clear the flag requests that are raised previously by setting corresponding bits in the MTOCIPCCLR register.

Out of the 32 MTOCIPC flags, the first four (bit 0 to bit 3) can generate corresponding MTOCIPC interrupts (MTOCIPCINT1 to MTOCIPCINT4) to the C28x PIE if enabled. The remaining 28 bits (bit 4 to bit 31) can be used as explained above in a software-based handshake (flag/ack) mechanism.

Note that the MTOCIPCSET registers are write-only by the M3 and will always read back as 0. The M3 should read the MTOCIPCFLG register to see pending requests. The MTOCIPCFLG and MTOCIPCSTS registers are read-only and will always reflect the current status of the corresponding IPC flag whether it has been requested or cleared. The MTOCIPCCLR and MTOCIPCACK bits are write-only and will always read back as 0.

1.12.4 CTOMIPC Communication

When a message is ready for the C28x to send to the M3, the C28x can notify that to the M3 using CTOMIPC flags or interrupts. The M3, upon polling for these CTOMIPC flags or upon getting a CTOMIPC interrupt, can respond to the message sent by the C28x. There are three 32-bit registers: CTOMIPCSET, CTOMIPCCLR and CTOMIPCFLG on the C28x memory map, and two 32-bit registers, CTOMIPCSTS and CTOMIPCACK, to implement the flag-based message communication from the C28x to the M3. The CTOMIPCSET register has 32 bits, each of which when set, is capable of enabling a corresponding flag bit in the CTOMIPCFLG register on the C28x and a status bit in the CTOMIPCSTS register on the M3.

For example,

1. When the C28x wants to notify a message readiness to the M3, the C28x can set bit 9 (IPC flag 10) in the CTOMIPCSET register.
2. This action of the C28x will result in setting bit 9 in the CTOMIPCSTS register on the M3 along with bit 9 in the CTOMIPCFLG register on the C28x. The CTOMIPCFLG register on the C28x and the CTOMIPCSTS register on the M3 are physically the same but named differently as the FLG register on the C28x and the STS register on the M3.
3. The M3 can poll for bit 9 (IPC flag 10) of the CTOMIPCSTS register while waiting for the message.
4. When bit 9 of the CTOMIPCSTS register gets set, the M3 can read the message if any or can take necessary action depending on the design of the user application.
5. Then, the M3 should acknowledge the C28x that it received the CTOMIPC request by setting bit 9 of the CTOMIPCACK register which will result in clearing bit 9 in the CTOMIPCFLG register and the CTOMIPCSTS register.

   Note that the CTOMIPCCLR register on the C28x and the CTOMIPCACK register on the M3 are physically the same, but named differently as the CLR register on the C28x and the ACK register on the M3.

For every bit in the CTOMIPCSET register, there is a bit in the CTOMIPCCLR register. The C28x can set bits in the CTOMIPCCLR register to clear corresponding flags (flag bit in CTOMIPCFLG and status bit in CTOMIPCSTS) that the C28x has set previously using the CTOMIPCSET register. Generally, the M3 should acknowledge a CTOMIPC request using the CTOMIPCACK register thus clearing the CTOMIPCFLG and CTOMIPCSTS bits. However, at times when the C28x did not receive any acknowledgement from the M3 about the reception of message in an expected duration, the C28x application software might want to clear the CTOMIPC flag request that it set earlier. In such a scenario, the C28x can clear the flag requests that it raised previously by setting the corresponding bits in the CTOMIPCCLR register.

Out of the 32 CTOMIPC flags, the first four bits (bit 0 to bit 3) can generate corresponding CTOMIPC interrupts (CTOMIPCINT1 to CTOMIPCINT4) to the M3 PIE if enabled. The remaining 28 bits (bit 4 to bit 31) can be used as explained above in a software-based handshake (flag/ack) mechanism.

Copyright © 2012–2019, Texas Instruments Incorporated

Note that the CTOMIPCSET registers are write-only by the C28x and will always read back as 0. The C28x should read the CTOMIPCFLG register to see pending requests. The CTOMIPCFLG and CTOMIPCSTS registers are read-only and will always reflect the current status of the corresponding IPC flag whether it has been requested or cleared. The CTOMIPCCLR and CTOMIPCACK bits are write-only and will always read back as 0. In the event of simultaneous write accesses to the same bits of the CTOMIPCACK and CTOMIPCSET register, the M3 CPU trying to clear and the C28 CPU trying to set, the C28 CPU receives priority.

1.12.5 Examples for Software IPC Procedure

The below are given suggested examples of the sequence to be followed in software for IPC.

1.12.5.1 IPC With Interrupts

Below is an example procedure for IPC usage when the M3 CPU wants to get some information from the C28x using MTOCIPCINT1:

- The M3 writes a ‘1’ in bit 0 of the MTOCIPCSET register and this generates the MTOCIPCINT0 to the C28x through the PIE.
- Bit 0 in the MTOCIPCFLG and MTOCIPCSTS registers get set. The C28x services the interrupt and in the corresponding ISR, the C28x loads the pre-defined information in the shared RAMs or CTOM-message RAM (user application has to define the ISR functionality).
- The C28x clears this MTOCIPC request by writing a ‘1’ to bit 0 of the MTOCIPCACK register at the end of the ISR.
- The M3 polls the status of bit 0 in the MTOCIPCFLG register and until the status is ‘1’, it understands that the C28x CPU has not serviced the interrupt. When the status becomes ‘0’, it understands that the C28x CPU has serviced the interrupt and reads the RAM from the predefined location and gathers the requested information.

1.12.5.2 IPC With Flags

Below is an example procedure for IPC usage when the C28x CPU wants to communicate a message to the M3 about a shared resource using CTOMIPC-flag 5:

- The C28x writes a ‘1’ to bit 5 of the CTOMIPCSET register to indicate that the M3 can go ahead and use a particular resource as the C28x is done with using that resource. (In user application software, IPC flag 6 will be tied to a particular resource and task.)
- When the M3 wants to use this resource, it will read bit 5 of the CTOMIPCSTS register. When the M3 reads bit 5 of the CTOMIPCSTS register as a ‘1’, it uses shared resource tied to IPC flag 6 in the pre-defined manner.
- After completing the task using the shared resource tied with IPC5, the M3 CPU will clear the flag by writing ‘1’ to bit 5 of CTOMIPCACK.
- When bit 5 of the CTOMIPCFLG register is a ‘1’ and the flag is not yet cleared by the M3 CPU, if the C28 CPU reads bit 5 of the CTOMIPCFLG register, the C28x reads it as a ‘1’. When the flag has been cleared by the M3, bit 5 of the CTOMIPCFLG will read ‘0’. Based on this, C28 software proceeds accordingly for its task.

1.12.6 IPC Message Registers

IPC message registers provide a simple and flexible way for the master subsystem and control subsystem to send messages between them. There are four dedicated IPC message registers on both subsystems. There is not any specific hardware definition for the usage of these registers. The user's application software has to define the usage of these registers. These registers can be used like mailboxes to send messages back and forth between the two subsystems when the software cannot use memories for inter processor communication. IPC Message registers on each of the subsystems are accessible by other subsystems.

IPC Message registers that the master subsystem can use to convey a message to the control subsystem are given in Table 1-36.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-36. MTOCIPC Message Registers

Register Name	Description	M3 Read/Write	C28x Read/Write
MTOCIPCCOM	MTOC IPC command register	Read/write	Read only
MTOCIPCADDR	MTOC IPC address register	Read/write	Read only
MTOCIPCDATAW	MTOC IPC Data write register	Read/write	Read only
MTOCIPCDATAR	MTOC IPC Data read register	Read only	Read/write

IPC Message registers that Control subsystem can use to convey a message to the master subsystem are given in Table 1-37.

Table 1-37. CTOMIPC Message Registers

Register Name	Description	M3 Read/Write	C28x Read/Write
CTOMIPCCOM	CTOM IPC command register	Read/write	Read only
CTOMIPCADDR	CTOM IPC address register	Read/write	Read only
CTOMIPCDATAW	CTOM IPC Data write register	Read/write	Read only
CTOMIPCDATAR	CTOM IPC Data read register	Read only	Read/write

Let us consider an example for the usage of these registers. The user's application might have a scenario where the M3 requires knowing the data in one of the memories mapped to the C28x to which the M3 does not have read access. For this scenario, software can be designed such that:

1. When the M3 requires the data from C28x memory, the M3 should write a value of 0x0000007 (arbitrarily represents "16-bit data read" command) in the MTOCIPCCOM register and the address where it wants the C28x to read data in the MTOCIPCADDR register.
2. After writing to these IPC message registers, the M3 should raise an IPC request using either an MTOCIPC flag or interrupt.
3. When the C28x gets this IPC request from the M3, C28x software should be designed to read the MTOCIPCCOM register and understand that the command value of 0x00000007 means 16-bit data command.
4. Upon understanding the command, C28x software should read the MTOCIPCADDR register to get the address that the M3 wrote.
5. Then the C28x software can read the data from that address and write the data that it obtained by reading that memory location in MTOCIPCDATAR for the M3 to read.

In this way, user application can define the usage of the IPC message registers to accomplish mailbox communications between the master and control subsystems.

1.12.7 Flash Pump Semaphore

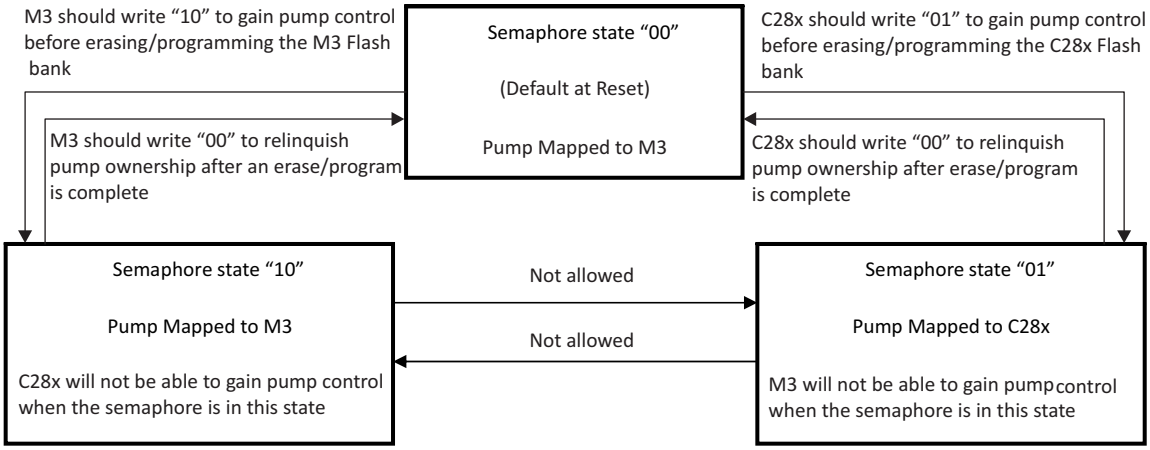
There are two flash banks in this device. One flash bank is in the master subsystem and another flash bank is in the control subsystem. The M3 core can erase, program, and read the master subsystem flash bank, and the C28x core can erase, program, and read the control subsystem flash bank. But both of these flash banks share a single flash pump for erase and program operations. Hence, only one core can perform erase/program operations on its flash bank at any given time.

When the M3 core is performing erase/program operations on M3 flash bank, the C28x core cannot erase/program the C28x flash bank, but the C28x core can execute code and/or read data from the C28x flash bank. Similarly, when the C28x core is performing erase/program operations on the C28x flash bank, the M3 core cannot erase/program the M3 flash bank, but the M3 core can execute code and/or read data from the M3 flash bank.

Copyright © 2012–2019, Texas Instruments Incorporated

As both banks are controlled by a single flash pump, the control signals to the pump will be controlled by either the C28x-Flash Module Controller or the M3-Flash Module Controller based on a hardware semaphore called flash pump semaphore (PUMPREQUEST). This semaphore can be accessed by both cores on their respective memory maps in the IPC register space. This flash pump semaphore is called MPUMPREQUEST on the M3 memory map and CPUMPREQUEST on the C28x memory map. Figure 1-26 shows flash pump allocation for different states of the flash pump semaphore (a 2-bit field called SEM in PUMPREQUEST registers).

Figure 1-26. Flash Pump Allocation for Different States of Flash Pump Semaphore



By default at reset, the flash pump is mapped to the M3 with a semaphore value of "00". Even though the pump is mapped to the M3 by default, the M3 should gain the pump by writing a value of "10" in the MPUMPREQUEST register when the M3 wants to erase/program the M3 flash bank. This prevents the C28x from grabbing the pump semaphore when the M3 is erasing/programming its bank. Once the M3 is done erasing/programming its bank, it should release the pump control by writing a value of "00" in to MPUMPREQUEST semaphore so that the C28x can grab pump control when needed.

Similarly, the C28x should gain the pump by writing a value of "01" in the CPUMPREQUEST register when the C28x wants to erase/program the C28x flash bank. This prevents the M3 from grabbing the pump semaphore when the C28x is erasing/programming its bank. Once the C28x is done erasing/programming its bank, it should release the pump control by writing a value of "00" to the CPUMPREQUEST semaphore so that the M3 can grab pump control when needed. In case of a conflict when both the M3 and C28x simultaneously try to gain pump control by writing appropriate values to the above registers, the M3 will gain pump control.

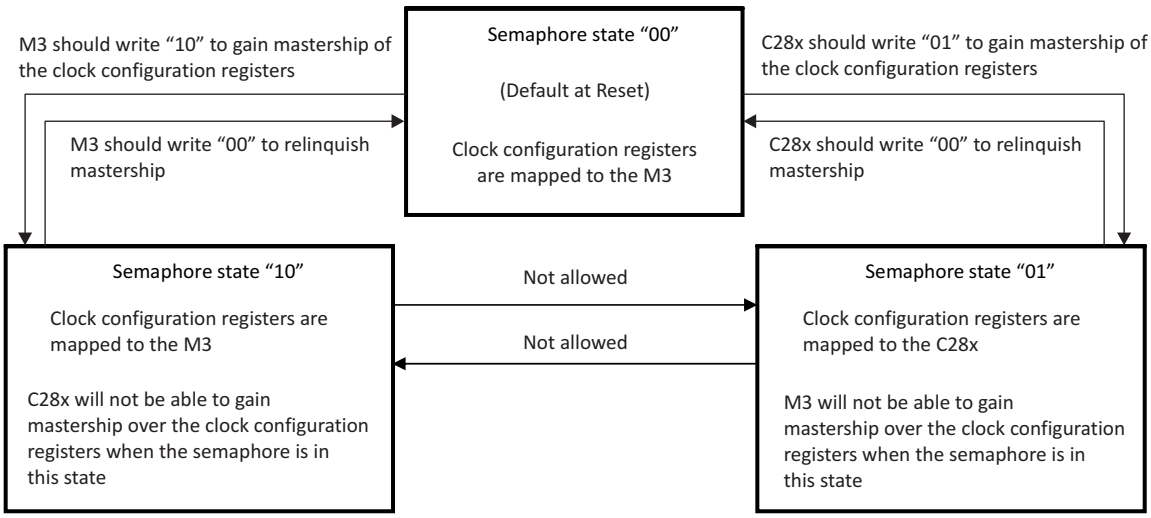
1.12.8 Clock Configuration Semaphore

The M3 has control over the device clock configuration registers by default. However, in order to allow the C28x core to erase/program its flash bank independent of the M3 core, a hardware clock configuration semaphore (CLKREQUEST) is provided so that the C28 CPU can get write access to configure the SYSPLLCTL, SYSPLLMULT, SYSPLLSTS, and SYSDIVSEL registers for the required frequency. This semaphore can be accessed by both cores on their respective memory maps in the IPC register space.

The clock configuration semaphore is called MCLKREQUEST on the M3 memory map and CCLKREQUEST on the C28x memory map. Figure 1-27 shows the mastership of clock configuration registers for different states of the clock configuration semaphore (a 2-bit field called SEM in CLKREQUEST registers).

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-27. Mastership of Clock Configuration Registers for Different States of Clock Configuration Semaphore



By default at reset, clock configuration registers are mapped to the M3 with a semaphore value of "00". Even though clock configuration registers are mapped to the M3 by default, the M3 should gain mastership by writing a value of "10" in the MCLKREQUEST register when the M3 wants to configure the clock settings and perform any operations at that clock frequency. This prevents the C28x from grabbing mastership over clock configuration registers when the M3 is configuring the clock settings and executing at that frequency.

Once the M3 is done with configuring the clock settings and any operations that it wants to perform at that configured frequency, it should release mastership of the clock configuration registers by writing a value of "00" in the MCLKREQUEST semaphore so that the C28x can request mastership over clock control registers when needed. If the user's application is designed in such a way that the C28x is not allowed to modify the clock settings and only the M3 configures the clock settings for the C28x as well (for flash erase/program or even for normal application execution), then it is suggested that the M3 should keep mastership over clock configuration registers by writing a value of "10" in the MCLKREQUEST register and not setting it back to "00".

If the user's application is designed in such a way that the C28x is allowed to configure the clock settings, then the C28x should gain mastership of the clock configuration registers by writing a value of "01" in the CCLKREQUEST register when the C28x wants to configure clock settings. This prevents the M3 from writing to clock configuration registers when the C28x is configuring them.

When C28x is configuring the Clock settings, the PLL has to be powered down before changing the PLL multiplier (if the PLL is being used in the application). PLL can be turned on once the PLL multiplier is configured. Below is the required sequence for PLL configuration from C28x:

1. Make C28x gain mastership of clock configuration registers using the clock configuration semaphore
2. Bypass the PLL
3. Power down the PLL (this is not required when configuring clocks from M3)
4. Maximize the clock dividers
5. Modify the PLL multiplier
6. Power on the PLL
7. Wait for the PLL to lock
8. Put the PLL back into the clock path
9. Modify the clock dividers as desired by the application

Copyright © 2012–2019, Texas Instruments Incorporated

Once the C28x is done with configuring the clock settings and performing any operations at that configured clock frequency, it should release mastership of the clock configuration registers by writing a value of "00" in the CCLKREQUEST semaphore so that the M3 can gain control when needed. In case of a conflict when both the M3 and C28x simultaneously try to gain control over the clock configuration registers by writing appropriate values to the above semaphore registers, the M3 will gain the mastership.

1.12.9 Free Running Counter

A 64-bit free running counter is present in the device and can be used to timestamp IPC events between processors. This 64-bit MIPCCOUNTERL(32-bit)/ MIPCCOUNTERH(32-bit) is clocked by the shared resources clock and is readable by the M3 and the C28x on their respective memory maps. These counter registers are reset to zero on reset.

Because the counter is 64-bits and the M3/C28x can only read 32-bits at a time, an issue can arise when reading the counters separately. If the low 32-bit counter is read just before it overflows and then the high 32-bit counter is read, the combined values read will be incorrect. To solve this, a snapshot for the high 32-bits counter is taken when a read is performed on the MIPCCOUNTERL register. When the M3/C28x reads the MIPCOUNTERH, the snapshot is fed back to the user instead of the current value in the MIPCOUNTERH register. Therefore, the user application software must always read MIPCCOUNTERL first and then read MIPCCOUNTERH.

The MIPCCOUNTERL/H is stopped only when both the M3 and C28 CPUs are halted (not the low power halt-mode). If either core is executing, the counter runs. It is suggested that the user application should disable interrupts when reading the counters.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13 System Control Registers

This section provides details of all the system control module registers that let users configure, control, and monitor various functions and features in the device.

Section 1.13.1 and Table 1-38 give details about the "base address+offset" of each register and identifies the reset source for each register and whether the register is accessible by the master subsystem or control subsystem or both. The below Register Map also provides details on whether the register is write protected and if the register is read-only.

Registers accessible by only the Cortex-M3 CPU in the master subsystem have only M3 "base address+offset" and registers accessible by only the C28x CPU in the control subsystem have only C28x "base address+offset" associated with it. The registers which are accessible by both have both M3 CPU and C28x CPU "base address + offset".

The "Write Once" or "Write = 1" in the Read-Only column of the below table means that the respective LOCK registers can be written only once and can only set the bits in the respective registers to "1." Writes of "0" are ignored and once set to "1" cannot be written again until the respective reset is generated to reset the state of the LOCK configuration.

Note that all the addresses in the memory map which are not defined in the below table are reserved. None of the reserved addresses, nor any of the reserved bits in registers should be written to. If there is a need to write to them they should be written back with the same bit value as read from them.

1.13.1 System Control, Configuration Register Map

Table 1-38. System Control, Configuration Registers Address Map

Register Acronym	Register Description	Size (x8)	C28 Offset (x16)	M3 Offset (x8)	C28 Protection	M3 Protection	Reset Source	Read Only
Master Subsystem Device Identification and System Control Registers:				0x400F:E000				
DID0	Device Identification Register 0	4		0x0			$\overline{\text{XRS}}$	
DID1	Device Identification Register 1	4		0x4			$\overline{\text{M3SYSRST}}$	Yes
DC1	Device Configuration 1 Register	4		0x10			$\overline{\text{XRS}}$	
DC2	Device Configuration 2 Register	4		0x14		MWRALLOW	$\overline{\text{XRS}}$	Yes
DC4	Device Configuration 4 Register	4		0x1C		MWRALLOW	$\overline{\text{XRS}}$	Yes
DC6	Device Configuration 6 Register	4		0x24		MWRALLOW	$\overline{\text{XRS}}$	Yes
DC7	Device Configuration 7 Register	4		0x28			$\overline{\text{XRS}}$	
SRCR0	Software Reset Control Register 0	4		0x40		MWRALLOW	$\overline{\text{M3SYSRST}}$	
SRCR1	Software Reset Control Register 1	4		0x44		MWRALLOW	$\overline{\text{M3SYSRST}}$	
SRCR2	Software Reset Control Register 2	4		0x48		MWRALLOW	$\overline{\text{M3SYSRST}}$	
SRCR3	Software Reset Control Register 3	4		0x4C		MWRALLOW	$\overline{\text{M3SYSRST}}$	
MRESC	Master Reset Cause Register	4		0x5C			$\overline{\text{POR}}$	
RCC	Run Mode Clock Configuration Register	4		0x060		MWRALLOW	$\overline{\text{M3SYSRST}}$	
GPIOHBCTL	Master GPIO High Performance Bus Control Register	4		0x6C		MWRALLOW	$\overline{\text{M3SYSRST}}$	
RCGC0	Run Mode Clock Gating Control Register 0	4		0x100		MWRALLOW	$\overline{\text{M3SYSRST}}$	

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-38. System Control, Configuration Registers Address Map (continued)

Register Acronym	Register Description	Size (x8)	C28 Offset (x16)	M3 Offset (x8)	C28 Protection	M3 Protection	Reset Source	Read Only
RCGC1	Run Mode Clock Gating Control Register 1	4		0x104		MWRALLOW	M3SYSRST	
RCGC2	Run Mode Clock Gating Control Register 2	4		0x108		MWRALLOW	M3SYSRST	
RCGC3	Run Mode Clock Gating Control Register 3	4		0x10C		MWRALLOW	M3SYSRST	
SCGC0	Sleep Mode Clock Gating Control Register 0	4		0x110		MWRALLOW	M3SYSRST	
SCGC1	Sleep Mode Clock Gating Control Register 1	4		0x114		MWRALLOW	M3SYSRST	
SCGC2	Sleep Mode Clock Gating Control Register 2	4		0x118		MWRALLOW	M3SYSRST	
SCGC3	Sleep Mode Clock Gating Control Register 3	4		0x11C		MWRALLOW	M3SYSRST	
DCGC0	Deep Sleep Mode Clock Gating Control Register 0	4		0x120		MWRALLOW	M3SYSRST	
DCGC1	Deep Sleep Mode Clock Gating Control Register 1	4		0x124		MWRALLOW	M3SYSRST	
DCGC2	Deep Sleep Mode Clock Gating Control Register 2	4		0x128		MWRALLOW	M3SYSRST	
DCGC3	Deep Sleep Mode Clock Gating Control Register 3	4		0x12C		MWRALLOW	M3SYSRST	
DSLPCLKCFG	Deep Sleep Clock Configuration Register	4		0x144		MWRALLOW	M3SYSRST	
DC10	Device Configuration 10 Register	4		0x194		MWRALLOW	XRS	Yes
Master Subsystem Device Identification and System Control Registers:				0x400F:B900				
MCNF	Master Subsystem Configuration	4		0x00		MWRALLOW	XRS	Yes
SERPLOOP	Serial Port Loop Back Control Register	4		0x08		MWRALLOW	M3SYSRST	
MCIBSTATUS	Master Subsystem: ACIB Status Register	4		0x0C		-	SRXRST	
MNMI Configuration Registers:				0x400F:BA00				
MNMICFG	M3NMI Configuration Register	4		0X0		MWRALLOW	M3SYSRST	
MNMIFLG	M3NMI Flag Register	4		0X4			XRS	
MNMIFLGCLR	M3NMI Flag Clear Register	4		0X8		MWRALLOW	M3SYSRST	
MNMIFLGFRC	M3NMI Flag Force Register	4		0X0C		MWRALLOW	M3SYSRST	
MNMIWDCNT	M3NMI Watchdog Counter Register	4		0X10		MWRALLOW	M3SYSRST	
MNMIWDPRD	M3NMI Watchdog Period Register	4		0X14		MWRALLOW	M3SYSRST	
M3 MWRALLOW Configuration Registers:				0x400F:B980			M3SYSRST	
MWRALLOW	M3 Configuration Write Allow Register	4		0x0		Privelege Mode	M3SYSRST	

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-38. System Control, Configuration Registers Address Map (continued)

Register Acronym	Register Description	Size (x8)	C28 Offset (x16)	M3 Offset (x8)	C28 Protection	M3 Protection	Reset Source	Read Only
MLOCK	M3 Configuration Lock Register	4		0x4		MWRALLOW	$\overline{\text{SRXRST}}$	READ ONLY, Write=1
Master Subsystem Reset Registers:				0x400F:B8C0				
CRESCNF	Control Subsystem: Reset Configuration/Control Register	4		0x0		MWRALLOW	$\overline{\text{XRS}}$	
CRESSTS	Control Subsystem: Reset Status Register	4		0x4		MWRALLOW	$\overline{\text{XRS}}$	
MWIR	Master Susystem: Wait-In-Reset Register	4		0xC		MWRALLOW	$\overline{\text{XRS}}$	
Device Low Power Mode Entry and Wake Up Register:				0x400F:B880				
CLPMSTAT	Control Subsystem: Low Power Mode Status Register	4		0x0			$\overline{\text{M3SYSRST}}$	
Clock Configuration Registers				0x400F:B800				
M3SSDIVSEL	Master Subsystem: Clock Divider Register	4		0X10		MWRALLOW	$\overline{\text{XRS}}$	
UPLLCTL	USB PLL Configuration Register	4		0X20		MWRALLOW	$\overline{\text{XRS}}$	
UPLLMULT	USB PLL Multiplier Register	4		0X24		MWRALLOW	$\overline{\text{XRS}}$	
UPLLSTS	USB PLL Lock Status Register	4		0X28			$\overline{\text{XRS}}$	
MCLKSTS	Missing Clock Status Register	4		0X30			$\overline{\text{XRS}}$	
MCLKFRCCLR	Missing Clock Force Register	4		0X38		MWRALLOW	$\overline{\text{XRS}}$	
MCLKEN	Missing Clock Enable Register	4		0X3C		MWRALLOW	$\overline{\text{XRS}}$	
MCLKLIMIT	Missing Clock Reference Limit Register	4		0X40		MWRALLOW	$\overline{\text{XRS}}$	
XPLLCLKCFG	XPLL CLKOUT Control Register	4		0X50		MWRALLOW	$\overline{\text{XRS}}$	
CCLKOFF	Control Subsystem: Clock Disable Register	4		0X60		MWRALLOW	$\overline{\text{XRS}}$	
CAN0BCLKSEL	Bit Clock Source Selection for CAN0 Register	4		0x70		MWRALLOW	$\overline{\text{M3SYSRST}}$	
CAN1BCLKSEL	Bit Clock Source Selection for CAN1 Register	4		0x74		MWRALLOW	$\overline{\text{M3SYSRST}}$	
Clock Configuration Registers			0x4400	0x400F:B800				
SYSPLLCTL	System PLL Configuration Register	4	0x0	0x00	EALLOW	MWRALLOW	$\overline{\text{XRS}}$	
SYSPLLMULT	System PLL Multiplier Register	4	0x2	0x04	EALLOW	MWRALLOW	$\overline{\text{XRS}}$	
SYSPLLSTS	System PLL Lock Status Register	4	0x4	0x08			$\overline{\text{XRS}}$	
SYSDIVSEL	System Clock Divider Register	4	0x6	0x0C	EALLOW	MWRALLOW	$\overline{\text{XRS}}$	
Control Subsystem Device Configuration Registers:			0x0886	0x400F:B900				

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-38. System Control, Configuration Registers Address Map (continued)

Register Acronym	Register Description	Size (x8)	C28 Offset (x16)	M3 Offset (x8)	C28 Protection	M3 Protection	Reset Source	Read Only
CCNF0	Control Subsystem: Peripheral Configuration 0 Register	4	0x00	0x10		MWRALLOW	$\overline{XRS}$	Yes
CCNF1	Control Subsystem: Peripheral Configuration 1 Register	4	0x02	0x14		MWRALLOW	$\overline{XRS}$	Yes
CCNF2	Control Subsystem: Peripheral Configuration 2 Register	4	0x04	0x18		MWRALLOW	$\overline{XRS}$	Yes
CCNF3	Control Subsystem: Peripheral Configuration 3 Register	4	0x06	0x1C		MWRALLOW	$\overline{XRS}$	Yes
CCNF4	C28 Flash Configuration Register	4	0x08	0x20		MWRALLOW	$\overline{XRS}$	Yes
Master Subsystem Memory Configuration Register:			0x08C0	0x400F:B900				
MEMCNF	Memory Configuration Register	4	0x00	0x30		MWRALLOW		Yes
Control Subsystem Device Configuration Register:			0x0880					
DEVICECNF	Device Configuration Register	4	0x0		EALLOW		$\overline{C28SYSRST}$, $\overline{M3SYSRST}$,$\overline{T}$ $\overline{RST}$, $\overline{XRS}$	
C28 REVID Register:			0x0882					
PARTID	Device Part ID Register	2	0x0				$\overline{XRS}$	Yes
REVID	Device Revision Register	2	0x1				$\overline{XRS}$	Yes
Control Subsystem Peripheral Clocking Control Registers:			0x7000					
CRESC	C28 Reset Cause Register	2	0x2C		EALLOW		$\overline{XRS}$	
CLKCTL	C28 CPU Timer 2 Clock Configuration Register	2	0x12		EALLOW		$\overline{XRS}$	
PCLKCR2	C28 Peripheral Clock Control Register 2	2	0X19		EALLOW		$\overline{C28SYSRST}$	
HISPCP	C28 High-Speed Clock Prescaler Register	2	0X1A		EALLOW		$\overline{C28SYSRST}$	
LOSPCP	C28 Low-Speed Clock Prescaler Register	2	0X1B		EALLOW		$\overline{C28SYSRST}$	
PCLKCR0	C28 Peripheral Clock Control Register 0	2	0X1C		EALLOW		$\overline{C28SYSRST}$	
PCLKCR1	C28 Peripheral Clock Control Register 1	2	0X1D		EALLOW		$\overline{C28SYSRST}$	
LPMCR0	C28 LPM Control Register 0	2	0X1E		EALLOW		$\overline{C28SYSRST}$	
PCLKCR3	C28 Peripheral Clock Control Register 3	2	0X20		EALLOW		$\overline{C28SYSRST}$	
CWIR	C28 Wait-In-Reset Register	2	0X2B		EALLOW		$\overline{XRS}$	
Control Subsystem NMI Configuration Registers:			0x7060					

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-38. System Control, Configuration Registers Address Map (continued)

Register Acronym	Register Description	Size (x8)	C28 Offset (x16)	M3 Offset (x8)	C28 Protection	M3 Protection	Reset Source	Read Only
CNMICFG	C28 NMI Configuration Register	2	0x00		EALLOW		$\overline{\text{XRS}}$	
CNMIFLG	C28 NMI Flag Register	2	0x01				$\overline{\text{C28SYSRST}}$	
CNMIFLGCLR	C28 NMI Flag Clear Register	2	0x02		EALLOW		$\overline{\text{C28SYSRST}}$	
CNMIFLGFRC	C28 NMI Flag Force Register	2	0x03		EALLOW		$\overline{\text{C28SYSRST}}$	
CNMIWDCNT	C28 NMI Watchdog Counter Register	2	0x04		EALLOW		$\overline{\text{C28SYSRST}}$	
CNMIWDPRD	C28 NMI Watchdog Period Register	2	0x05		EALLOW		$\overline{\text{C28SYSRST}}$	
Control Susbystem XINT Registers			0x7070					
XINT1CR	C28 XINT1 Configuration Register	2	0x00				$\overline{\text{C28SYSRST}}$	
XINT2CR	C28 XINT2 Configuration Register	2	0x01				$\overline{\text{C28SYSRST}}$	
XINT3CR	C28 XINT3 Configuration Register	2	0x02				$\overline{\text{C28SYSRST}}$	
XINT1CTR	C28 XINT1 Counter Register	2	0x08				$\overline{\text{C28SYSRST}}$	
XINT2CTR	C28 XINT2 Counter Register	2	0x09				$\overline{\text{C28SYSRST}}$	
XINT3CTR	C28 XINT3 Counter Register	2	0x0A				$\overline{\text{C28SYSRST}}$	
Master Subsystem CSM Configuration Registers				0x400F: B400				
Z1_CSMKEY0	128-bit CSMKEY Register for M3 Zone1	4		0x0			$\overline{\text{M3SYSRST}}$	
Z1_CSMKEY1		4		0x4			$\overline{\text{M3SYSRST}}$	
Z1_CSMKEY2		4		0x8			$\overline{\text{M3SYSRST}}$	
Z1_CSMKEY3		4		0xC			$\overline{\text{M3SYSRST}}$	
Z1_ECSLKEY0	64-bit ECSLKEY Register for M3 Zone1	4		0x10			$\overline{\text{M3SYSRST}}$	
Z1_ECSLKEY1		4		0x14			$\overline{\text{M3SYSRST}}$	
Z2_CSMKEY0	128-bit CSMKEY Register for M3 Zone2	4		0x18			$\overline{\text{M3SYSRST}}$	
Z2_CSMKEY1		4		0x1C			$\overline{\text{M3SYSRST}}$	
Z2_CSMKEY2		4		0x20			$\overline{\text{M3SYSRST}}$	
Z2_CSMKEY3		4		0x24			$\overline{\text{M3SYSRST}}$	
Z2_ECSLKEY0	64-bit ECSLKEY Register for M3 Zone2	4		0x28			$\overline{\text{M3SYSRST}}$	
Z2_ECSLKEY1		4		0x2C			$\overline{\text{M3SYSRST}}$	
Z1_CSMCR	Status & Control Register for M3 Zone1	4		0x80			$\overline{\text{M3SYSRST}}$	
Z2_CSMCR	Status & Control Register for M3 Zone2	4		0x84			$\overline{\text{M3SYSRST}}$	
Z1_GRABSECTR	Grab Flash-Sectors Register for M3 Zone1	4		0x90			$\overline{\text{M3SYSRST}}$	Yes
Z1_GRABRAMR	Grab RAM-blocks Register for M3 Zone1	4		0x94			$\overline{\text{M3SYSRST}}$	Yes
Z2_GRABSECTR	Grab Flash-Sectors Register for M3 Zone2	4		0x98			$\overline{\text{M3SYSRST}}$	Yes

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-38. System Control, Configuration Registers Address Map (continued)

Register Acronym	Register Description	Size (x8)	C28 Offset (x16)	M3 Offset (x8)	C28 Protection	M3 Protection	Reset Source	Read Only
Z2_GRABRAMR	Grab RAM-blocks Register for M3 Zone1	4		0x9C			$\overline{\text{M3SYSRST}}$	Yes
Z1_EXEONLYR	Execute-Only Select Register for flash sectors mapped to M3 Zone1	4		0xB0			$\overline{\text{M3SYSRST}}$	Yes
Z2_EXEONLYR	Execute-Only Select Register for flash sectors mapped to M3 Zone2	4		0xB4			$\overline{\text{M3SYSRST}}$	Yes
OTPSECLOCK	OTPSECLOCK Register	4		0x120			$\overline{\text{M3SYSRST}}$	Yes
Control Subsystem CSM Configuration Registers			0x0AE0					
CSMKEY0	128-bit CSMKEY Register for C28x	4	0x0				$\overline{\text{C28SYSRST}}$	
CSMKEY1		4	0x2				$\overline{\text{C28SYSRST}}$	
CSMKEY2		4	0x4				$\overline{\text{C28SYSRST}}$	
CSMKEY3		4	0x6				$\overline{\text{C28SYSRST}}$	
CSMCR	Status & Control Register for C28x	2	0xF				$\overline{\text{C28SYSRST}}$	
ECSLKEY0		4	0x10				$\overline{\text{C28SYSRST}}$	
ECSLKEY1		4	0x12				$\overline{\text{C28SYSRST}}$	
EXEONLYR	Grab Flash-Sectors Register for C28x	2	0x15				$\overline{\text{C28SYSRST}}$	Yes
µCRC Configuration Register				0x400FB600				
µCRCCONFIG	µCRC Configuration Register	4		0x0			$\overline{\text{M3SYSRST}}$	
µCRCCONTROL	µCRC Control Register	4		0x4			$\overline{\text{M3SYSRST}}$	
µCRCRES	µCRC Result Register	4		0x8			$\overline{\text{M3SYSRST}}$	
Master Subsystem IPC Registers			0x4E00	0x400F:B700				
CTOMIPCACK	C28 to M3 core IPC request acknowledge register	4		0x00			$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	
CTOMIPCSTS	C28 to M3 core IPC request status register	4		0x04			$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	
MTOCIPCSET	M3 to C28 core IPC request set register	4		0x08			$\overline{\text{SRXRST}}$	
MTOCIPCCLR	M3 to C28 core IPC request clear register	4		0x0C			$\overline{\text{SRXRST}}$	
MTOCIPCFLG	M3 to C28 core IPC request flag register	4		0x10			$\overline{\text{SRXRST}}$	
Reserved	Reserved	4		0x14				
MIPCCOUNTERL	M3 IPC Counter Low Register (clocked by shared resource clock)	4	0x0C	0x18			$\overline{\text{SRXRST}}$	Yes
MIPCCOUNTERH	M3 IPC Counter High Register (clocked by shared resource clock)	4	0x0E	0x1C			$\overline{\text{SRXRST}}$	Yes
CTOMIPCCOM	C28 to M3 IPC Command Register	4	0x10	0x20			$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	Yes for M3
CTOMIPCADDR	C28 to M3 IPC Address Register	4	0x12	0x24			$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	Yes for M3

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-38. System Control, Configuration Registers Address Map (continued)

Register Acronym	Register Description	Size (x8)	C28 Offset (x16)	M3 Offset (x8)	C28 Protection	M3 Protection	Reset Source	Read Only
CTOMIPCDATAW	C28 to M3 IPC Data Write Register	4	0x14	0x28			$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	Yes for M3
CTOMIPCDATAR	C28 to M3 IPC Data Read Register	4	0x16	0x2C			$\overline{\text{SRXRST}}$	
MTOCIPCCOM	M3 to C28 IPC Command Register	4	0x18	0x30			$\overline{\text{SRXRST}}$	
MTOCIPCADDR	M3 to C28 IPC Address Register	4	0x1A	0x34			$\overline{\text{SRXRST}}$	
MTOCIPCDATAW	M3 to C28 IPC Data Write Register	4	0x1C	0x38			$\overline{\text{SRXRST}}$	
MTOCIPCDATAR	M3 to C28 IPC Data Read Register	4	0x1E	0x3C			$\overline{\text{C28RST}}$, $\overline{\text{SRXRST}}$	Yes for M3
CTOMIPCBOOTSTS	IPC Boot Status Register	4	0x20	0x40			$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	Yes for M3
MTOCIPCBOOTMODE	IPC Boot Mode Register	4	0x22	0x44			$\overline{\text{SRXRST}}$	
MPUMPREQUEST	Flash PUMP semaphore M3 request register	4		0x48		MWRALLOW	$\overline{\text{SRXRST}}$	
MCLKREQUEST	Clock configuration semaphore M3 request register	4		0x4C		MWRALLOW	$\overline{\text{SRXRST}}$	
Reserved	Reserved	48		0x50				
Control Subsystem IPC Registers			0x4E00	0x400F:B700				
CTOMIPCSET	C28 to M3 core IPC request set register	4	0x00				$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	
CTOMIPCCLR	C28 to M3 core IPC request clear register	4	0x02				$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	
CTOMIPCFLG	C28 to M3 core IPC request flag register	4	0x04				$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	
MTOCIPCACK	M3 to C28 core IPC request acknowledge register	4	0x06				$\overline{\text{SRXRST}}$	
MTOCIPCSTS	M3 to C28 core IPC request status register	4	0x08				$\overline{\text{SRXRST}}$	
Reserved	Reserved	4	0xA					
MIPCCOUNTERL	M3 IPC Counter Low Register (clocked by shared resource clock)	4	0x0C	0x18			$\overline{\text{SRXRST}}$	Yes
MIPCCOUNTERH	M3 IPC Counter High Register (clocked by shared resource clock)	4	0x0E	0x1C			$\overline{\text{SRXRST}}$	Yes
CTOMIPCCOM	C28 to M3 IPC Command Register	4	0x10	0x20			$\overline{\text{C28SYSRST}}$ $\overline{\text{SRXRST}}$	
CTOMIPCADDR	C28 to M3 IPC Address Register	4	0x12	0x24			$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	
CTOMIPCDATAW	C28 to M3 IPC Data Write Register	4	0x14	0x28			$\overline{\text{C28RST}}$	
CTOMIPCDATAR	C28 to M3 IPC Data Read Register	4	0x16	0x2C			$\overline{\text{SRXRST}}$	Yes for C28x
MTOCIPCCOM	M3 to C28 IPC Command Register	4	0x18	0x30			$\overline{\text{SRXRST}}$	Yes for C28x

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-38. System Control, Configuration Registers Address Map (continued)

Register Acronym	Register Description	Size (x8)	C28 Offset (x16)	M3 Offset (x8)	C28 Protection	M3 Protection	Reset Source	Read Only
MTOCIPCADDR	M3 to C28 IPC Address Register	4	0x1A	0x34			$\overline{\text{SRXRST}}$	Yes for C28x
MTOCIPCDATAW	M3 to C28 IPC Data Write Register	4	0x1C	0x38			$\overline{\text{SRXRST}}$	Yes for C28x
MTOCIPCDATAR	M3 to C28 IPC Data Read Register	4	0x1E	0x3C			$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	
CTOMIPCBOOTSTS	IPC Boot Status Register	4	0x20	0x40			$\overline{\text{C28SYSRST}}$, $\overline{\text{SRXRST}}$	
MTOCIPCBOOTMODE	IPC Boot Mode Register	4	0x22	0x44			$\overline{\text{SRXRST}}$	Yes for C28x
CPUMPREQUEST	Flash programming semaphore PUMP request register	4	0x24		EALLOW	MWRALLOW	$\overline{\text{SRXRST}}$	
CCLKREQUEST	Clock configuration semaphore C28 request register	4	0x26		EALLOW	MWRALLOW	$\overline{\text{SRXRST}}$	
Reserved	Reserved	48	0x28					

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.2 Device Identification and Device Configuration

1.13.2.1 Device Identification 0 (DID0) Register

Figure 1-28. Device Identification 0 (DID0) Register

31	30 28	27 24	23 16
Rsvd	VER	Reserved	CLASS
R-0	R-0x2	R-0	R-0x40

15 0
REVID
R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-39. Device Identification 0 (DID0) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved		Reserved
30-28	VER	0x2	DID0 Version Version of DID0 register used is 0x2.
27-24	Reserved		Reserved
23-16	CLASS	0x40	F28M35x
15-0	REVID		Current Rev. ID of the device

1.13.2.2 Device Identification 1 (DID1) Register

Figure 1-29. Device Identification 1 (DID1) Register

31 28	27 24	23 16
VER	FAM	PARTNO
R-0x1	R-0x1	R-x

15 13	12 8	7 5	4 3	2	1 0
PINCOUNT	Reserved	TEMP	PACKAGE	ROHS	QUAL
R-0x4	R-0	R-0	R-0x1	R-0x1	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; x= indeterminate

Table 1-40. Device Identification 1 (DID1) Register Field Descriptions

Bit	Field	Value	Description
31-28	VER	0x1	DID1 Version Version of DID1 register used is 0x1.
27-24	FAM	0x1	Concerto device family
23-16	PARTNO		Part Number These 8 bits specify the actual device part number. This register field is replicated on the C28 side in the PARTID register.
15-13	PINCOUNT		Package Pin Count
		0x4	144 pins
12-8	Reserved		Reserved
7-5	TEMP		Temperature Range
		2	Extended Industrial Temperature Range (-40°C to 105°C).
4-3	PACKAGE		Package Type
		0x1	LQFP

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-40. Device Identification 1 (DID1) Register Field Descriptions (continued)

Bit	Field	Value	Description
2	ROHS		ROHS Compliance
		0	Package is not ROHS compliant.
		1	Package is ROHS compliant. Concerto devices are ROHS compliant.
1-0	QUAL		Qual Status This field specifies the qualification status of the device. The QUAL field value is encoded as follows:
		0	Engineering sample (TMX)
		1	Pilot production (TMP)
		2	Fully qualified (TMS)
			All other encodings are reserved.

1.13.2.3 Device Configuration 1 (DC1) Register

Figure 1-30. Device Configuration 1 (DC1) Register

31	29	28	27	16
Reserved		WDT1	Reserved	
R-x		R-x	R-x	

15	5	4	3	2	1	0
Reserved		PLL	WDT0	Reserved		JTAG
R-x		R-x	R-x	R-x		R-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; x= indeterminate

Table 1-41. Device Configuration 1 (DC1) Register Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
28	WDT1		Watchdog Timer1
		0	WDT1 is not present
		1	WDT1 is present
27-5	Reserved		Reserved
4	PLL		PLL
		0	PLL is not present
		1	PLL is present
3	WDT0		Watchdog Timer0
		0	WDT0 is not present
		1	WDT0 is present
2-1	Reserved		Reserved
0	JTAG		JTAG Debugger Interface
		0	JTAG debugger interface is not present
		1	JTAG debugger interface is present

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.2.4 Device Configuration 2 (DC2) Register

Figure 1-31. Device Configuration 2 (DC2) Register

31	30	29	28	27	26	25	24
Reserved	EPI	Reserved					
R-x	R-x	R-x					R-x

23	22	21	20	19	18	17	16
Reserved				TIMER3	TIMER2	TIMER1	TIMER0
R-x				R-x	R-x	R-x	R-x

15	14	13	12	11	10	9	8
Reserved	I2C1	Reserved	I2C0	Reserved			
R-x	R-x	R-x	R-x	R-x			

7	6	5	4	3	2	1	0
SSI3	SSI2	SSI1	SSI0	UART3	UART2	UART1	UART0
R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-42. Device Configuration 2 (DC2) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved		Reserved
30	EPI		EPI Whether EPI is present or not depends on the device configuration.
		0	EPI is not present
		1	EPI is present
29-20	Reserved		Reserved
19	TIMER3		GPT Timer3 Whether GPT3 is present or not depends on the device configuration.
		0	GPT3 is not present
		1	GPT3 is present
18	TIMER2		GPT Timer2 Whether GPT2 is present or not depends on the device configuration.
		0	GPT2 is not present
		1	GPT2 is present
17	TIMER1		GPT Timer1 Whether GPT1 is present or not depends on the device configuration.
		0	GPT1 is not present
		1	GPT1 is present
16	TIMER0		GPT Timer0 Whether GPT0 is present or not depends on the device configuration.
		0	GPT0 is not present
		1	GPT0 is present
15	Reserved		Reserved
14	I2C1		I2C1 Whether I2C1 is present or not depends on the device configuration.
		0	I2C1 is not present
		1	I2C1 is present
13	Reserved		Reserved
12	I2C0		II2C0 Whether I2C0 is present or not depends on the device configuration.
		0	I2C0 is not present
		1	I2C0 is present

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-42. Device Configuration 2 (DC2) Register Field Descriptions (continued)

Bit	Field	Value	Description
11-8	Reserved		Reserved
7	SSI3		SSI3 Whether SSI3 is present or not depends on the device configuration.
		0	SSI3 is not present
		1	SSI3 is present
6	SSI2		SSI2 Whether SSI2 is present or not depends on the device configuration.
		0	SSI2 is not present
		1	SSI2 is present
5	SSI1		SSI1 Whether SSI1 is present or not depends on the device configuration.
		0	SSI1 is not present
		1	SSI1 is present
4	SSI0		SSI0 Whether SSI0 is present or not depends on the device configuration.
		0	SSI0 is not present
		1	SSI0 is present
3	UART3		UART3 Whether UART3 is present or not depends on the device configuration.
		0	UART3 is not present
		1	UART3 is present.
2	UART2		UART2 Whether UART2 is present or not depends on the device configuration.
		0	UART2 is not present
		1	UART2 is present.
1	UART1		UART1 Whether UART1 is present or not depends on the device configuration.
		0	UART1 is not present
		1	UART1 is present.
0	UART0		UART0 Whether UART0 is present or not depends on the device configuration.
		0	UART0 is not present
		1	UART0 is present.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.2.5 Device Configuration 4 (DC4) Register

Figure 1-32. Device Configuration 4 (DC4) Register

31	30	29	28	27	26	25	24
Reserved			EMAC0	Reserved			E1588
R-x			R-x	R-x			R-x

23							16
Reserved							
R-x							

15	14	13	12	11	10	9	8
Reserved		µDMA	ROM	Reserved			GPIOJ
R-x				R-x			

7	6	5	4	3	2	1	0
GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-43. Device Configuration 4 (DC4) Register Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
28	EMAC0		Ethernet MAC Layer 0 Whether EMAC0 is present or not depends on the device configuration.
		0	EMAC0 is not present
		1	EMAC0 is present
27-25	Reserved		Reserved
24	E1588		1588-Capable EMAC 0 Whether 1588-capable EMAC0 is present or not depends on the device configuration.
		0	EMAC0 is not 1588 capable
		1	EMAC0 is 1588 capable
23-14	Reserved		Reserved
13	µDMA		µDMA Whether µDMA is present or not depends on the device configuration.
		0	µDMA is not present
		1	µDMA is present
12	ROM		On-Chip M3 Code ROM
		0	M3 code ROM is not present
		1	M3 code ROM is present
11-9	Reserved		Reserved
8	GPIOJ		GPIO PortJ Whether GPIOJ is present or not depends on the device configuration.
		0	GPIO PortJ is not present
		1	GPIO PortJ is present
7	GPIOH		GPIO PortH Whether GPIOH is present or not depends on the device configuration.
		0	GPIO PortH is not present
		1	GPIO PortH is present
6	GPIOG		GPIO PortG Whether GPIOG is present or not depends on the device configuration.
		0	GPIO PortG is not present
		1	GPIO PortG is present

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-43. Device Configuration 4 (DC4) Register Field Descriptions (continued)

Bit	Field	Value	Description
5	GPIOF		GPIO PortF Whether GPIOF is present or not depends on the device configuration.
		0	GPIO PortF is not present
		1	GPIO PortF is present
4	GPIOE		GPIO PortE Whether GPIOE is present or not depends on the device configuration.
		0	GPIO PortE is not present
		1	GPIO PortE is present
3	GPIOD		GPIO PortD Whether GPIOD is present or not depends on the device configuration.
		0	GPIO PortD is not present
		1	GPIO PortD is present
2	GPIOC		GPIO PortC Whether GPIOC is present or not depends on the device configuration.
		0	GPIO PortC is not present
		1	GPIO PortC is present
1	GPIOB		GPIO PortB Whether GPIOB is present or not depends on the device configuration.
		0	GPIO PortB is not present
		1	GPIO PortB is present
0	GPIOA		GPIO PortA Whether GPIOA is present or not depends on the device configuration.
		0	GPIO PortA is not present
		1	GPIO PortA is present

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.2.6 Device Configuration 6 (DC6) Register

Figure 1-33. Device Configuration 6 (DC6) Register

31	16
Reserved	
R-x	

15	5	4	3	2	1	0
Reserved		USBPHY0	Reserved		USB0	
R-x		R-x	R-x		R-x	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; x= indeterminate

Table 1-44. Device Configuration 6 (DC6) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4	USBPHY0		USB0 PHY Whether USBPHY0 is present or not depends on the device configuration.
		0	USB0 PHY is not present
		1	USB0 PHY is present
3-2	Reserved		Reserved
1-0	USB0		USB0 Controller Functionality USB functionality allowed in the device, depends on device configuration.
		0x00	No USB functionality
		0x01	Device Only
		0x10	Device or Host
		0x11	OTG

1.13.2.7 Device Configuration 10 (DC10) Register

Figure 1-34. Device Configuration 10 (DC10) Register

31	26	25	24	23	16
Reserved		CAN1	CAN0	Reserved	
R-x		R-x	R-x	R-x	

15	1	0
Reserved		UART4
R-x		R-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; x= indeterminate

Table 1-45. Device Configuration 10 (DC10) Register Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25	CAN1		CAN1 Whether CAN1 is present or not depends on the device configuration.
		0	CAN1 is not present
		1	CAN1 is present
24	CAN0		CAN0 Whether CAN0 is present or not depends on the device configuration.
		0	CAN0 is not present
		1	CAN0 is present
23-1	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-45. Device Configuration 10 (DC10) Register Field Descriptions (continued)

Bit	Field	Value	Description
0	UART4		UART4 Whether UART4 is present or not depends on the device configuration. UART4 is not present UART4 is present

1.13.2.8 Device Configuration 7 (DC7) Register

This register can be used to verify µDMA channel features. A "1" indicates that the channel is available on this device. In this device all the 32 channels of µDMA are available in all device configurations.

Figure 1-35. Device Configuration 7 (DC7) Register

31	0
DMACH	
R-0xFFFFFFFF	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-46. Device Configuration 7 (DC7) Register Field Descriptions

Bit	Field	Value	Description
31-0	DMACH		DMA Channel Present All the DMA channels [31:0] of the µDMA are available in this device.

1.13.2.9 Master Subsystem Configuration (MCNF) Register

The Master Subsystem Configuration (MCNF) register is shown and described in the figure and table below.

Figure 1-36. Master Subsystem Configuration (MCNF) Register

31	16
Reserved	
R-0	

15	9	8	7	3	2	0
Reserved		µCRC	Reserved		FLASH	
R-0		R-x	R-0		R-x	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; x= indeterminate

Table 1-47. Master Subsystem Configuration (MCNF) Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	µCRC		µCRC Configuration
		0	µCRC is disabled
		1	µCRC is present
7-3	Reserved		Reserved
2-0	FLASH		M3 Flash Size Configuration
		111	512KB
		110	256KB

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.2.10 Serial Port Loop Back Control (SERPLOOP) Register

Figure 1-37. Serial Port Loop Back Control (SERPLOOP) Register

31	16
Reserved	
R-0	

15	9	8	7	2	1	0
Reserved		UART4 TOSCIA	Reserved		SSI3TOSPIA	
R-0		R/W-0	R-0		R/W-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-48. Serial Port Loop Back Control (SERPLOOP) Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	UART4TOSCIA		UART4-to-SCIA Loopback UART4 to SCIA connection logic control
		0	Not connected (default on reset)
		1	UART4 connected to SCIA
7-2	Reserved		Reserved
1-0	SSI3TOSPIA		SSI3-to-SPIA Loopback SSI3 to SPIA internal connection logic control
		0 x	Not Connected (default on reset)
		1 0	SSI3 Connected to SPI-A (SPI-A is in Slave Mode)
		1 1	SSI3 Connected to SPI-A (SPI-A is in Master Mode)

1.13.2.11 Master Subsystem: ACIB Status (MCIBSTATUS) Register

Figure 1-38. Master Subystem: ACIB Status (MCIBSTATUS) Register

15	8
CIBBUSCLKCNT	
R-0:0	

7	3	2	1	0
Reserved		INTS	READY	APGOODSTS
R-0:0		R-x	R-x	R-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; x= indeterminate

Table 1-49. Master Subsystem: ACIB Status (MCIBSTATUS) Register Field Descriptions

Bit	Field	Value	Description
15-8	CIBBUSCLKCNT		8-bit CIBBUSCLK Counter This free running 8-bit counter is incremented by the CIBBUSCLK. If the counter overflows, it will rewind to zero and continue counting. This counter is used to indicate if CIBBUSCLK is present.
7-3	Reserved		Reserved
2	INTS		INTS signal state Reading this bit will give the current state of the INTS CIB signal at the input.
1	READY		READY signal state Reading this bit will give the current state of the READY CIB signal at the input.
0	APGOODSTS		Analog System Power Good Status Reading this bit gives the power state of the Analog Subsystem.
		0	Analog subsystem power not present
		1	Analog subsystem power present

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.2.12 C28 Device Part ID (PARTID) Register

Figure 1-39. C28 Device Part ID (PARTID) Register

15	8	7	0
PARTTYPE		PARTNO	
R-0x0		R	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-50. C28 Device Part ID (PARTID) Register Field Descriptions

Bit	Field	Value	Description
15-8	PARTTYPE		Device Part Type These 8-bits are hard wired and indicate the device memory type
		0x00	Flash Device
		0xFF	ROM Device
7-0	PARTNO		Device Part Number These 8-bits specify the actual device part number. These bits are configured via OTP. On reset, the M3 boot software will read the part number from the OTP and load it into the PARTNO field of the DID1 register. This register field is a replica on the C28 side of the PARTNO bits of the DID1 register.

1.13.2.13 C28 Revision ID (REVID) Register

Figure 1-40. C28 Revision ID (REVID) Register

15	0
REVID	
R-0x0 (first silicon rev)	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-51. C28 Revision ID (REVID) Register Field Descriptions

Bit	Field	Value	Description
15-0	REVID		Silicon Revision Number These 16-bits specify the silicon revision number for the particular part.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.2.14 Control Subsystem Device Configuration (DEVICECNF) Register (EALLOW Protected)

Figure 1-41. Control Subsystem Device Configuration (DEVICECNF) Register

31	24
Reserved	
R-0	

23	20	19	18	16
Reserved		ENPROT		Reserved
R-0		R/W-1		R-1,1,1

15	8
Reserved	Reserved
R/W-0	R-0:0

7	6	5	4	3	2	1	0
Reserved	Reserved	$\overline{\text{XRS}}$	Reserved	VMAPS	Reserved	Reserved	
R/W-0	R-1	R-0/1	R-0	R-0/1	R-0	R/W-1,1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-52. Control Subsystem Device Configuration (DEVICECNF) Register Field Descriptions

Bit	Field	Value	Description
31-20	Reserved		Reserved
19	ENPROT		Enable Write-Read Protection Enable write-read protection mode bit.
18-6	Reserved		Reserved
5	XRSn		Input XRSn Status Reset input signal status.
4	Reserved		Reserved
3	VMAPS		VMAP Status VMAP configure status.
2-0	Reserved		Reserved

1.13.2.15 Control Subsystem Peripheral Configuration 0 (CCNF0) Register

Figure 1-42. Control Subsystem Peripheral Configuration 0 (CCNF0) Register

31	16
Reserved	
R-0:0	

15	13	12	11	10	9	8
Reserved		MCBSP	Reserved	SCI	Reserved	SPI
R-0:0		R-x	R-0	R-x	R-0:0	R-x

7	5	4	3	1	0
Reserved		I2C	Reserved		HRPWM
R-0:0		R-x	R-0:0		R-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-53. Control Subsystem Peripheral Configuration 0 (CCNF0) Register Field Descriptions

Bit	Field	Value	Description
31-13	Reserved		Reserved
12	MCBSP		McBSP Module Configuration When set, this enables the control subsystem McBSP module.
		0	McBSP module is disabled
		1	McBSP module is enabled
11	Reserved		Reserved
10	SCI		SCI Module Configuration When set, this enables the control subsystem SCI module.
		0	SCI module is disabled
		1	SCI module is enabled
9	Reserved		Reserved
8	SPI		SPI Module Configuration When set, this enables the control subsystem SPI module.
		0	SPI module is disabled
		1	SPI module is enabled
7-5	Reserved		Reserved
4	I2C		I2C Module Configuration When set, this enables the control subsystem I2C module.
		0	I2C module is disabled
		1	I2C module is enabled
3-1	Reserved		Reserved
0	HRPWM		HRPWM Module Configuration When set, this enables the HRPWM module.
		0	HRPWM module is disabled
		1	HRPWM module is enabled

1.13.2.16 Control Subsystem Peripheral Configuration 1 (CCNF1) Register

Figure 1-43. Control Subsystem Peripheral Configuration 1 (CCNF1) Register

31															16
Reserved															
R-0:0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EQEP2	EQEP1	ECAP6	ECAP5	ECAP4	ECAP3	ECAP2	ECAP1	EPWM8	EPWM7	EPWM6	EPWM5	EPWM4	EPWM3	EPWM2	EPWM1
R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-54. Control Subsystem Peripheral Configuration 1 (CCNF1) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-14	EQEP2-1		eQEP2-1 Configuration When set, this enables the respective eQEP module.
		0	Respective eQEP module is disabled
		1	Respective eQEP module is enabled
13-8	ECAP6-1		eCAP6-1 Configuration When set, this enables the respective eCAP module.
		0	Respective eCAP module is disabled
		1	Respective eCAP module is enabled

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-54. Control Subsystem Peripheral Configuration 1 (CCNF1) Register Field Descriptions (continued)

Bit	Field	Value	Description
7-0	EPWM8-1		ePWM8-1 Configuration When set, this enables the respective ePWM module.
		0	Respective ePWM module is disabled
		1	Respective ePWM module is enabled

1.13.2.17 Control Subsystem Peripheral Configuration 2 (CCNF2) Register

Figure 1-44. Control Subsystem Peripheral Configuration 2 (CCNF2) Register

31	16
Reserved	
R-0:0	

15 9	8	7 1	0
Reserved	EQEP3	Reserved	EPWM9
R-0:0	R-x	R-0	R-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-55. Control Subsystem Peripheral Configuration 2 (CCNF2) Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	EQEP3		eQEP3 Configuration. When set, this enables the eQEP3 module.
		0	eQEP3 module is disabled
		1	eQEP3 module is enabled
7-1	Reserved		Reserved
0	EPWM9		ePWM9 Configuration. When set, this enables the ePWM9 module.
		0	ePWM9 module is disabled
		1	ePWM9 module is enabled

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.2.18 Control Subsystem Peripheral Configuration 3 (CCNF3) Register

Figure 1-45. Control Subsystem Peripheral Configuration 3 (CCNF3) Register

31	12	11	10	0
Reserved		C28DMA	Reserved	
R-0		R-x	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-56. Control Subsystem Peripheral Configuration 3 (CCNF3) Register Field Descriptions

Bit	Field	Value	Description
31-12	Reserved		Reserved
11	C28DMA		C28DMA Configuration When set, this enables the C28 DMA module.
		0	C28 DMA module is disabled
		1	C28DMA module is enabled
10-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.2.19 Control Subsystem Peripheral Configuration 4 (CCNF4) Register

Figure 1-46. Control Subsystem Peripheral Configuration 4 (CCNF4) Register

31	16
Reserved	
R-x	

15	3	2	0
Reserved		FLASH	
R-x			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-57. Control Subsystem Peripheral Configuration 4 (CCNF4) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2-0	FLASH		C28 Flash Size Configuration
		111	512KB
		110	256KB

1.13.2.20 Master Subsystem Memory Configuration (MEMCNF) Register

NOTE: This register is read only from the control subsystem.

Figure 1-47. Master Subsystem Memory Configuration (MEMCNF) Register

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		S7	S6	S5	S4	S3	S2	S1	S0
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-58. Master Subsystem Memory Configuration (MEMCNF) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	S7-0		S7-0 Shared Memory Configuration S7-0 Shared RAM configuration.
		0	S7-0 RAM is disabled
		1	S7-0 RAM is enabled

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.3 Reset Control and Status Registers

1.13.3.1 Subsystem Reset Configuration/Control (CRESCNF) Register

Figure 1-48. Subsystem Reset Configuration/Control (CRESCNF) Register

31	18	17	16
Reserved		$\overline{\text{ACIBRESET}}$	M3RSnIN
R-0:0		R/W-0	R/W-0

15	1	0
Reserved		Reserved
R-0		R-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-59. Subsystem Reset Configuration/Control (CRESCNF) Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved		Reserved
17	$\overline{\text{ACIBRESET}}$		M3 Reset to ACIB
		0	$\overline{\text{ACIBRESET}}$ to ACIB is low, causing a ACIB reset (holds Analog Subsystem in reset).
		1	$\overline{\text{ACIBRESET}}$ to ACIB is high, enabling ACIB operation (allowing Analog Subsystem to run).
16	M3RSnIN		M3 Reset to C28 CPU
		0	RSn to C28 CPU is low, causing a C28 CPU and C28 subsystem reset.
		1	RSn to C28 CPU is high, bringing out the C28 CPU and subsystem out of reset.
15-0	Reserved		Reserved

1.13.3.2 Control Subsystem Reset Status (CRESSTS) Register

Figure 1-49. Control Subsystem Reset Status (CRESSTS) Register

31	19	18	17	16
Reserved		CHWBISTRST		CNMIWDRST
R-0		R/W-1		R/W-1

15	1	0
Reserved		CRES
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-60. Control Subsystem Reset Status (CRESSTS) Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved		Reserved
18-17	CHWBISTRST		C28 Reset Cause Flag – set by hardware when C28 HWBIST controller fired a reset to the C28 CPU; this flag does not indicate fail conditions – fail conditions cause an NMI to the M3 and C28 CPUs
		00	C28 CPU was not reset due to C28 HW BIST reset
		11	C28 CPU was reset due to C28 HW BIST reset
			This status bit is a latched flag This flag can be cleared by the M3 CPU by writing a “1”
			Note: This flag is 2 bits to take care of any error conditions that may cause inadvertent setting of a single bit flag;

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-60. Control Subsystem Reset Status (CRESSTS) Register Field Descriptions (continued)

Bit	Field	Value	Description
16	CNMIWDRST		C28 Reset Cause Flag – set by hardware when C28 NMIWD fired a reset to the C28 CPU and subsystem
		0	C28 CPU was not reset by the C28 NMIWD
		1	C28 CPU was reset due to C28 NMIWD reset
			This status bit is a latched flag
			This flag can be cleared by the M3 CPU by writing a “1”
15-1	Reserved		Reserved
0	CRES		C28 Reset Status Bit
		0	C28 CPU is in reset
		1	C28 CPU is out of reset

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.3.3 Master Reset Cause (MRESC) Register

Figure 1-50. Master Reset Cause (MRESC) Register

31	30	29	28	27	26	25	24
ACIBERRNMI	C28NMIWDRST	M3BISTERRNMI	C28BISTERRNMI	C28PIENMI	HWBIST		EXTGPIO
R-0	R/W-0	R/w-0	R/w-0	R/W-0	R/w-0		R/W-0

23	17	16
Reserved		MCLKNMI
R-0		R/W-0

15	8
Reserved	
R-0:0	

7	6	5	4	3	2	1	0
Reserved		WDT1	SW	WDT0	Reserved	POR	$\overline{\text{XRS}}$
R-0:0		R/W-0	R/W-0	R/W-0	R-0:0	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-61. Master Reset Cause (MRESC) Register Field Descriptions

Bit	Field	Value	Description
31	ACIBERRNMI		ACIB Error NMI Unserviced
		0	No M3 NMIWD device reset since the previous POR. Writing a "0" to this bit clears it.
		1	CIB access signals stuck condition caused an NMI which was not serviced by the CPU and that caused an NMIWD reset.
30	C28NMIWDRST		C28 NMIWD Reset NMI Unserviced
		0	No M3 NMIWD device reset since the previous POR. Writing a "0" to this bit clears it.
		1	C28 NMIWD errored or timed out and triggered a M3 NMI and this triggered the M3 NMIWD. M3 did not respond to the NMI and the M3 NMIWD fired a reset.
29	M3BISTERRNMI		M3 BIST Error NMI Unserviced
		0	M3 did not respond to the NMI and the NMIWD fired a reset. If ‘0’ then there was NMIWD event that caused a device reset since the previous POR Writing a ‘0’ to this bit clears this bit
		1	If set, indicates that M3 hardware BIST errored or timed out and triggered a M3 NMI and this triggered the M3 NMIWD
28	C28BISTERRNMI		C28 BIST Error NMI Unserviced
		0	M3 did not respond to the NMI and the NMIWD fired a reset. If ‘0’ then there was NMIWD event that caused a device reset since the previous POR Writing a ‘0’ to this bit clears this bit
		1	If set, indicates that C28 hardware BIST errored or timed out and triggered a M3 NMI and this triggered the M3 NMIWD.
27	C28PIENMI		C28 PIE NMI Error Unserviced
		0	No M3 NMIWD event that caused a device reset since the previous POR. Writing a "0" to this bit clears it.
		1	C28 PIE NMI vector fetch error triggered a M3 NMI and this triggered the M3 NMIWD. M3 did not respond to the NMI and the NMIWD fired a reset.
26-25	HWBIST		HWBIST Unserviced. Can be used by the boot ROM to decide to jump to appropriate code
		0	If ‘00’ then there was no M3 HWBIST reset issued to the M3 CPU. Writing a ‘00’ to these bits clears these bits
		1	If set to 11, indicates that M3 CPU HWBIST has run to completion and issued a reset to the M3 CPU.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-61. Master Reset Cause (MRESC) Register Field Descriptions (continued)

Bit	Field	Value	Description
24	EXTGPIO		External GPIO NMI Unserviced
		0	NMIWD event caused a device reset since the previous POR. Writing a "0" to this bit clears it.
		1	External NMI input triggered a M3 NMI and this triggered the M3 NMIWD. M3 did not respond to the NMI and the NMIWD fired a reset.
23-17	Reserved		Reserved
16	MCLKNMI		Missing Clock Condition NMI Unserviced
		0	No NMIWD event that caused a device reset since the previous POR. Writing a "0" to this bit clears it.
		1	Missing clock detection triggered a M3 NMI and this triggered the M3 NMIWD. M3 did not respond to the NMI and the NMIWD fired a reset.
15-6	Reserved		Reserved
5	WDT1		Watchdog Timer 1 Reset
		0	No WDT1 time out reset causing device reset since the previous POR. Writing a "0" to this bit clears it.
		1	M3 WDT1 timed out and caused a device reset.
4	SW		Software NVIC Reset
		0	No NVIC software reset request that caused a device reset since the previous POR. Writing a "0" to this bit clears it.
		1	SW reset request from the NVIC SYSRESETREQ register caused a device reset.
3	WDT0		Watchdog Timer 0 Reset
		0	No WDT0 time out reset fired since the previous POR. Writing a "0" to this bit clears it.
		1	M3 WDT0 timed out and caused a device reset.
2	Reserved		Reserved
1	POR		POR Reset
		0	No POR reset fired indicating a POR condition. Writing a "0" to this bit clears it.
		1	Power-on reset caused a device reset.
0	$\overline{\text{XRS}}$		External Reset Input
		0	No external reset input since the previous POR. Writing a "0" to this bit clears it.
		1	External reset input pin caused a device reset.

1.13.3.4 C28 Reset Cause Register (CRESC) Register

Figure 1-51. C28 Reset Cause Register (CRESC) Register

15	5	4	3	2	1	0
Reserved		C28_NMIRSN	POR	$\overline{\text{XRS}}$	Reserved	
R-0		R/W-0	R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-62. C28 Reset Cause Register (CRESC) Register Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	1	Reserved
4	C28_NMIRSN		C28 NMI WDOG reset. If set, indicates that C28 NMI WDOG caused the reset. If ‘0’ then there was no C28 NMI WDOG reset since the previous POR
		0	Clears this bit.
		1	No effect

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-62. C28 Reset Cause Register (CRESC) Register Field Descriptions (continued)

Bit	Field	Value	Description
3	POR		POR reset. If set, indicates that a power-on reset caused a device reset. If '0' then POR reset fired indicating a POR condition.
		0	Clears this bit
		1	No effect
2	$\overline{\text{XRS}}$		External Reset Input. A '1' indicates that an external reset input pin caused a device reset. If '0' then there was no external reset input since the previous POR.
		0	Clears this bit
		1	No effect
1-0	Reserved		Reserved

1.13.3.5 Software Reset Control 0 (SRCR0) Register

NOTE: Writes to this register are masked by the DC1 register.

Putting the module into reset and bringing it out of reset is done by software. When a particular bit is set, the module goes into reset and to bring the module out of reset, software has to again write a '0' explicitly to the register.

Figure 1-52. Software Reset Control 0 (SRCR0) Register

31	29	28	27	16
Reserved		WDT1	Reserved	
R-0:0		R/W-0	R-0:0	

15	4	3	2	0
Reserved		WDT0	Reserved	
R-0:0		R/W-0	R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-63. Software Reset Control 0 (SRCR0) Register Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
28	WDT1		WDT1 S/W 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.
27-4	Reserved		Reserved
3	WDT0		WDT0 S/W 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		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.3.6 Software Reset Control 1 (SRCR1) Register

NOTE: Writes to this register are masked by the DC2 register.

Putting the module into reset and bringing it out of reset is done by software. When a particular bit is set, the module goes into reset and to bring the module out of reset, software has to again write a '0' explicitly to the register.

Figure 1-53. Software Reset Control 1 (SRCR1) Register

31	30	29	28	27	26	25	24
Reserved	EPI	Reserved					
R-0	R/W-0	R-0					
23	22	21	20	19	18	17	16
Reserved				TIMER3	TIMER2	TIMER1	TIMER0
R-0				R/W-0	R/W-0:0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
Reserved	I2C1	Reserved	I2C0	Reserved			
R-0	R/W-0	R-0	R/W-0	R-0			
7	6	5	4	3	2	1	0
SSI3	SSI2	SSI1	SSI0	UART3	UART2	UART1	UART0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-64. Software Reset Control 1 (SRCR1) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved		Reserved
30	EPI		EPI S/W Reset Control When this bit is set, EPI 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.
29-20	Reserved		Reserved
19	TIMER3		TIMER3 S/W Reset Control When this bit is set, GPT3 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		TIMER2 S/W Reset Control When this bit is set, GPT2 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		TIMER1 S/W Reset Control When this bit is set, GPT1 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		TIMER0 S/W Reset Control When this bit is set, GPT0 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		Reserved
14	I2C1		I2C1 S/W Reset Control When this bit is set, I2C1 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		Reserved
12	I2C0		I2C0 S/W Reset Control When this bit is set, I2C0 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.
11-8	Reserved		Reserved
7	SSI3		SSI3 S/W Reset Control When this bit is set, SSI3 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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-64. Software Reset Control 1 (SRCR1) Register Field Descriptions (continued)

Bit	Field	Value	Description
6	SSI2		SSI2 S/W Reset Control When this bit is set, SSI2 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	SSI1		SSI1 S/W Reset Control When this bit is set, SSI1 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		SSI0 S/W Reset Control When this bit is set, SSI0 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	UART3		UART3 S/W Reset Control When this bit is set, UART3 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	UART2		UART2 S/W Reset Control When this bit is set, UART2 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		UART1 S/W Reset Control When this bit is set, UART1 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		UART0 S/W Reset Control When this bit is set, UART0 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.

1.13.3.7 Software Reset Control 2 (SRCR2) Register

NOTE: Writes to this register are masked by the DC4 register.

Putting the module into reset and bringing it out of reset is done by software. When a particular bit is set, the module goes into reset and to bring the module out of reset, software has to again write a '0' explicitly to the register.

Figure 1-54. Software Reset Control 2 (SRCR2) Register

31	30	29	28	27	26	25	24
Reserved			EMAC0	Reserved			
R-0			R/W-0	R-0			

23	22	21	20	19	18	17	16
Reserved							USB
R-0							R/W-0

15	14	13	12	11	10	9	8
Reserved		µDMA	Reserved				GPIOJ
R-0		R/W-0	R-0				R/W-0

7	6	5	4	3	2	1	0
GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-65. Software Reset Control 2 (SRCR2) Register Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
28	EMAC0		EMAC0 S/W Reset Control When this bit is set, EMAC is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
27-17	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-65. Software Reset Control 2 (SRCR2) Register Field Descriptions (continued)

Bit	Field	Value	Description
16	USB		USB S/W Reset Control When this bit is set, USB 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		Reserved
13	µDMA		µDMA S/W Reset Control When this bit is set, µDMA 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.
12-9	Reserved		Reserved
8	GPIOJ		GPIOJ SW Reset Control When this bit is set, GPIOJ is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
7	GPIOH		GPIOH SW Reset Control When this bit is set, GPIOH is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
6	GPIOG		GPIOG SW Reset Control When this bit is set, GPIOG 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		GPIOF SW Reset Control When this bit is set, GPIOF 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		GPIOE SW Reset Control When this bit is set, GPIOE 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		GPIOF SW Reset Control When this bit is set, GPIOD 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		GPIOC SW Reset Control When this bit is set, GPIOC 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		GPIOB SW Reset Control When this bit is set, GPIOB 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		GPIOA SW Reset Control When this bit is set, GPIOA 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.

1.13.3.8 Software Reset Control 3 (SRCR3) Register

NOTE: Writes to this register are masked by the DC10 register.

Putting the module into reset and bringing it out of reset is done by software. When a particular bit is set, the module goes into reset and to bring the module out of reset, software has to again write a '0' explicitly to the register.

Figure 1-55. Software Reset Control 3 (SRCR3) Register

31	26	25	24	23	16
Reserved		CAN1	CAN0	Reserved	
R-0:0		R/W-0	R/W-0	R-0:0	

15	1	0
Reserved		UART4
R-0:0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-66. Software Reset Control 3 (SRCR3) Register Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25	CAN1		CAN1 S/W Reset Control When this bit is set, CAN01 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.
24	CAN0		CAN0 S/W Reset Control When this bit is set, CAN10 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.
23:1	Reserved		Reserved
0	UART4		UART4 S/W Reset Control When this bit is set, UART4 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.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.4 WIRMODE Registers

1.13.4.1 Master Subystem Wait-In-Reset (MWIR) Register

Figure 1-56. Master Subsystem Wait-In-Reset (MWIR) Register

31	3	2	1	0
Reserved		SAMPLE	EMU1	EMU0
R/W-0:0		R/W-0	R/W-pin state	R/W-pin state

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-67. Master Subsystem Wait-In-Reset (MWIR) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	SAMPLE		Re-sample EMU0 and EMU1 Pins
		0	Has no effect
		1	Forces a re-sample of the EMU0 and EMU1 pins and the values will be latched in the EMU0/EMU1 bits.
1	EMU1		Latched State of EMU1 Pin The state of EMU1 pin is latched on reset ($\overline{XRS}$ AND POR) or when the SAMPLE bit is triggered. Reading this bit will give the state of the EMU1 pin on reset or when sampled.
		0	Has no effect
		1	Forces the bit to "1"
0	EMU0		Latched State of EMU0 Pin The state of EMU0 pin is latched on reset ($\overline{XRS}$ AND POR) or when the SAMPLE bit is triggered. Reading this bit will give the state of the EMU0 pin on reset or when sampled.
		0	Has no effect
		1	Forces the bit to "1"

1.13.4.2 C28 Wait-In-Reset (CWIR) Register

Figure 1-57. C28 Wait-In-Reset (CWIR) Register

15	3	2	1	0
Reserved		SAMPLE	EMU1	EMU0
R/W-0:0		R/W-0	R/W-pin state	R/W-pin state

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-68. C28 Wait-In-Reset (CWIR) Register Field Descriptions

Bit	Field	Value	Description
15-3	Reserved		Reserved
2	SAMPLE		Re-sample EMU0 and EMU1 Pins
		0	Has no effect
		1	Forces a r-sample of the EMU0 and EMU1 pins and the values will be latched in the EMU0/EMU1 bits.
1	EMU1		Latched State of EMU1 Pin The state of EMU1 pin is latched on reset ($\overline{XRS}$ AND POR) or when the SAMPLE bit is triggered. Reading this bit will give the state of the EMU1 pin on reset or when sampled.
		0	Has no effect
		1	Forces the bit to "1"

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-68. C28 Wait-In-Reset (CWIR) Register Field Descriptions (continued)

Bit	Field	Value	Description
0	EMU0		Latched State of EMU0 Pin The state of EMU0 pin is latched on reset ($\overline{\text{XRS}}$ AND POR) or when the SAMPLE bit is triggered. Reading this bit will give the state of the EMU0 pin on reset or when sampled.
		0	Has no effect
		1	Forces the bit to "1"

1.13.5 Exception and Interrupts

1.13.5.1 M3NMI Configuration (MNMICFG) Register

NOTE: Clearing all latched NMI error flag conditions will stop the NMI watchdog counter and reset it. Any new error condition that is latched will restart the counter.

The user should clear the NMI interrupt flag and clear all flags together to generate a new interrupt if a new error event occurs.

Figure 1-58. M3NMI Configuration (MNMICFG) Register

31				16
Reserved				
R-0				
15	**10**	**9**	**8**	**1**	**0**
Reserved		ACIBERRE	Reserved		NMIE
R-0		R/W-0	R-0		R-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-69. M3NMI Configuration (MNMICFG) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Reserved
9	ACIBERRE		ACIBERR NMI Enable This bit enables the ACIBERR NMI condition to generate an NMI interrupt to the M3 CPU. Once enabled, the bit cannot be cleared by the user. Only a device reset causing M3 reset will clear the bit. Writes of 0 are ignored. Reading the bit will indicate if the ACIBERR NMI is enabled or disabled. **Note:** The ACIBERR NMI condition needs to be disabled at reset.
		0	ACIBERR NMI disabled
		1	ACIBERR NMI enabled
8-1	Reserved		Reserved
0	NMIE		NMI Enable This bit is always set to '1', meaning any NMI condition will generate an NMI interrupt to the M3 CPU and kick off the NMI watchdog counter. The bit cannot be cleared by the user.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.5.2 M3NMI Flag (MNMIFLG) Register

Figure 1-59. M3NMI Flag (MNMIFLG) Register

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

15	14	13	12	11	10	9	8
Reserved						ACIBERR	C28NMIWDRST
R-0						R-0	R-0

7	6	5	4	3	2	1	0
C28PIENMIERR	EXTGPIO	C28BISTERR	M3BISTERR	Reserved		CLOCKFAIL	NMIINT
R-0	R-0	R/W-1	R/W-1	R-0:0		R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-70. M3NMI Flag (MNMIFLG) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Reserved
9	ACIBERR		CIB Error NMI Flag This bit indicates if there is a stuck condition on the CIB INTS or CIB READY signals causing an NMI condition. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an $\overline{\text{XRS}}$ reset.
		0	No CIB INTS or READY stuck error condition pending
		1	CIB INTS or READY stuck error condition generated
8	C28NMIWDRST		C28 NMI WD Reset Flag This bit indicates that a reset was issued by the C28 NMIWD since the C28 did not service its NMI interrupt. Once enabled, the flag cannot be cleared by the user. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an $\overline{\text{XRS}}$ reset.
		0	No C28 NMI WD RST condition pending
		1	C28 NMI WD RST condition generated
7	C28PIENMIERR		C28 PIE NMIERR NMI Flag This bit indicates that an error condition was generated during NMI vector fetch from C28 PIE. Once enabled, the flag cannot be cleared by the user. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an $\overline{\text{XRS}}$ reset.
		0	No C28 PIE NMIERR condition pending
		1	C28 PIE NMIERR condition generated
6	EXTGPIO		External GPIO NMI Flag This bit indicates if the external GPIO pin generated an NMI interrupt. The bit can only be cleared by the user writing to the respective bit in the NMIFLGCLR register or by an $\overline{\text{XRS}}$ reset.
		0	No external GPIO NMI pending
		1	External GPIO NMI condition pending
5	C28BISTERR		HW BIST Error NMI Flag: This bit indicates if the time out error or a signature mismatch error condition during hardware BIST of M3 occurred. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset.
		0	No C28 HWBIST error condition pending
		1	C28 BIST error condition generated
4	M3BISTERR		HW BIST Error NMI Flag: This bit indicates if the time out error or a signature mismatch error condition during hardware BIST of M3 occurred. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset.
		0	No M3 HWBIST error condition pending
		1	M3 BIST error condition generated
1	CLOCKFAIL		Clock Fail NMI Flag This bit indicates if the CLOCKFAIL condition is latched. These bits can only be cleared by the user writing to the respective bit in the NMIFLGCLR register or by an $\overline{\text{XRS}}$ reset.
		0	No CLOCKFAIL condition pending
		1	CLOCKFAIL condition generated

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-70. M3NMI Flag (MNMIFLG) Register Field Descriptions (continued)

Bit	Field	Value	Description
0	NMIINT		NMI Interrupt Flag This bit indicates if an M3 NMI interrupt was generated. This bit can only be cleared by the user writing to the respective bit in the NMIFLGCLR register or by an $\overline{XRS}$ reset. No further NMI interrupts pulses are generated until this flag is cleared by the user.
		0	No NMI interrupt generated
		1	NMI interrupt generated

1.13.5.3 M3NMI Flag Clear (MNMIFLGCLR) Register

NOTE: If hardware is trying to set a bit to "1" while software is trying to clear a bit to "0" on the same cycle, hardware has priority.

Users should clear the pending FAIL flag first and then clear the NMIINT flag.

Figure 1-60. M3NMI Flag Clear (MNMIFLGCLR) Register

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

15	14	13	12	11	10	9	8
Reserved						ACIBERR	C28NMIWDRST
R-0						R/W-0	R/W-0

7	6	5	4	3	2	1	0
C28PIENMIERR	EXTGPIO	C28BISTERR	M3BISTERR	Reserved		CLOCKFAIL	NMIINT
R/W-0	R/W-0	R/W-0	R/W-0	R-0:0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-71. M3NMI Flag Clear (MNMIFLGCLR) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Reserved
9	ACIBERR		CIB Error NMI Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.
8	C28NMIWDRST		C28 NMI WD Reset Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.
7	C28PIENMIERR		C28 PIE NMIERR NMI Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.
6	EXTGPIO		External GPIO NMI Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.
5	C28BISTERR		C28 BIST Error Flag
		0	Writes of 0 are ignored. Always reads back 0.
		1	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Note 1: If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. Note 2: Users should clear the pending FAIL flag first and then clear the NMIINT flag.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-71. M3NMI Flag Clear (MNMIFLGCLR) Register Field Descriptions (continued)

Bit	Field	Value	Description
4	M3BISTERR		M3 BIST Error Flag
		0	Writes of 0 are ignored. Always reads back 0.
		1	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Note 1: If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. Note 2: Users should clear the pending FAIL flag first and then clear the NMIINT flag.
3-2	Reserved		Reserved
1	CLOCKFAIL		Clock Fail NMI Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.
0	NMIINT		NMI Interrupt Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.

1.13.5.4 M3NMI Flag Force (MNMIFLGFRC) Register

Figure 1-61. M3NMI Flag Force (MNMIFLGFRC) Register

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

15	14	13	12	11	10	9	8
Reserved						ACIBERR	C28NMIWDRST
R-0						R/W-0	R/W-0

7	6	5	4	3	2	1	0
C28PIENMIERR	EXTGPIO	C28BISTERR	M3BISTERR	Reserved		CLOCKFAIL	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R-0:0		R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-72. M3NMI Flag Force (MNMIFLGFRC) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Reserved
9	ACIBERR		CIB Error NMI Flag Force
		0	Ignored; always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Sets the corresponding flag bit in the NMIFLG register.
8	C28NMIWDRST		C28 NMI WD Reset Flag Force
		0	Ignored; always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Sets the corresponding flag bit in the NMIFLG register.
7	C28PIENMIERR		C28 PIE NMIERR NMI Flag Force
		0	Ignored; always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Sets the corresponding flag bit in the NMIFLG register.
6	EXTGPIO		External GPIO NMI Flag Force
		0	Ignored; always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Sets the corresponding flag bit in the NMIFLG register.
5	C28BISTERR		C28 BIST Error Flag
		0	Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-72. M3NMI Flag Force (MNMIFLGFRC) Register Field Descriptions (continued)

Bit	Field	Value	Description
4	M3BISTERR		M3 BIST Error Flag.
		0	Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers.
3-2	Reserved		Reserved
1	CLOCKFAIL		Clock Fail NMI Flag Force
		0	Ignored; always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Sets the corresponding flag bit in the NMIFLG register.
0	Reserved		Reserved

1.13.5.5 M3NMI Watchdog Counter (MNMIWDCNT) Register

Figure 1-62. M3NMI Watchdog Counter (MNMIWDCNT) Register

31	16	15	0
Reserved		NMIWDCNT	
R-0:0		R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-73. M3NMI Watchdog Counter (MNMIWDCNT) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	NMIWDCNT		NMI Watchdog Counter This 16-bit incremental counter will start incrementing whenever any one of the enabled "NMI" flags are set. If the counter reaches the period value, an $\overline{\text{NMIRS}}$ signal is fired which will then reset the full device. See Section 1.3 for more details on the reset behavior. The counter will reset to zero when it reaches the period value and will then restart counting if any of the enabled "NMI" flags are set. Normally, the software would respond to the NMI interrupt generated and clear the offending FLAG(s) before the NMI watchdog triggers a reset. In some situations, the software may decide to allow the watchdog to reset the device anyway. If no enabled "NMI" flag is set, then the counter will reset to zero and remain at zero until an enabled "NMI" flag is set. The counter is clocked at the M3 SSCLK rate.

1.13.5.6 M3NMI Watchdog Period (MNMIWDPRD) Register

Figure 1-63. M3NMI Watchdog Period (MNMIWDPRD) Register

31	16	15	0
Reserved		NMIWDPRD	
R-0:0		R/W-0xFFFF	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-74. M3NMI Watchdog Period (MNMIWDPRD) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	NMIWDPRD		M3 NMI Watchdog Period This 16-bit value contains the period value at which a reset is generated when the watchdog counter matches. At reset, this value is set at the maximum. The software can decrease the period value at initialization time. Writing a PERIOD value that is smaller then the current counter value will automatically force an $\overline{\text{NMIRS}}$ to the M3 and hence reset the watchdog counter.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.5.7 C28 NMI Configuration (CNMICFG) Register

NOTE: Clearing all latched NMI error flag conditions will stop the NMI watchdog counter and reset it. Any new error condition that is latched will restart the counter.

The user should clear the NMI interrupt flag and clear all flags together to generate a new interrupt if a new error event occurs.

Figure 1-64. C28 NMI Configuration (CNMICFG) Register

15	7	6	5	4	3	1	0
Reserved		ACIBERRE	C28BISTERR	M3BISTERR	Reserved		CNMIE
R-0:0		R/W-0	R/W-1	R/W-1	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-75. C28 NMI Configuration (CNMICFG) Register Field Descriptions

Bit	Field	Value	Description
15-7	Reserved		Reserved
6	ACIBERRE		ACIBERR NMI Enable This enables the ACIBERR NMI condition to generate an NMI interrupt to the C28 CPU. Once enabled, the bit cannot be cleared by the user. Only a device reset causing C28 SYSRSn reset will clear the bit. Writes of 0 are ignored. Reading the bit will indicate if the ACIBERR NMI is enabled or disabled. **Note:** The ACIBERR NMI condition needs to be disabled at reset.
		0	ACIBERR NMI disabled
		1	ACIBERR NMI enabled
5	C28BISTERR		HW BIST Error NMI Flag: This bit indicates if the time out error or a signature mismatch error condition during hardware BIST of C28 occurred. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset:
		0	No C28 HWBIST error condition pending
		1	C28 BIST error condition generated
4	M3BISTERR		HW BIST Error NMI Flag: This bit indicates if the time out error or a signature mismatch error condition during hardware BIST of M3 occurred. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset:
		0	No M3 HWBIST error condition pending
		1	M3 BIST error condition generated
3-1	Reserved		Reserved
0	CNMIE		NMI Enable This bit is always set to "1" meaning any NMI condition will generate an NMI interrupt to the C28 CPU and kick off the NMI watchdog counter. the bit cannot be cleared by the user.

1.13.5.8 C28 NMI Flag (CNMIFLG) Register

NOTE: The C28NMIFLG register is only reset by the $\overline{\text{XRS}}$ signal and not the C28 $\overline{\text{SYSRS}}$ signal. This is so the cause of the particular reset condition can be identified.

Figure 1-65. C28 NMI Flag (CNMIFLG) Register

15	8
Reserved	
R-0:0	

7	6	5	4	3	2	1	0
Reserved	ACIBERR	C28BISTERR	M3BISTERR	C28FLUNCERR	C28RAMUNCERR	CLOCKFAIL	NMIINT
R-0:0	R-0	R/W-0	R/W-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-76. C28 NMI Flag (CNMIFLG) Register Field Descriptions

Bit	Field	Value	Description
15-7	Reserved		Reserved
6	ACIBERR		CIB Error NMI Flag This bit indicates if there is a stuck condition on the CIB INTS or CIB READY signals causing an NMI condition. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an $\overline{XRS}$ reset.
		0	No CIB INTS or READY stuck error condition pending
		1	CIB INTS or READY stuck error condition generated
5	C28BISTERR		C28 BIST Error Flag
		0	Writes of 0 are ignored. Always reads back 0.
		1	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Note 1: If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. Note 2: Users should clear the pending FAIL flag first and then clear the NMIINT flag.
4	M3BISTERR		M3 BIST Error Flag
		0	Writes of 0 are ignored. Always reads back 0.
		1	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Note 1: If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. Note 2: Users should clear the pending FAIL flag first and then clear the NMIINT flag.
3	C28FLUNCERR		C28 Flash Uncorrectable Error NMI Flag This bit indicates if the C28 flash uncorrectable error condition is latched. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an $\overline{XRS}$ reset.
		0	No C28 Flash uncorrectable error condition pending
		1	C28 Flash uncorrectable error condition generated
2	C28RAMUNCERR		C28 RAM Uncorrectable Error NMI Flag This bit indicates if an uncorrectable error occurred on a C28 RAM access and that condition is latched. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an $\overline{XRS}$ reset.
		0	No C28 RAM uncorrectable error condition pending
		1	C28 RAM uncorrectable error condition generated
1	CLOCKFAIL		Clock Fail NMI Flag This bit indicates if the CLOCKFAIL condition is latched. These bits can only be cleared by the user writing to the respective bit in the NMIFLGCLR register or by an $\overline{XRS}$ reset.
		0	No CLOCKFAIL condition pending
		1	CLOCKFAIL condition generated
0	NMIINT		NMI Interrupt Flag his bit indicates if an NMI interrupt was generated. This bit can only be cleared by the user writing to the respective bit in the NMIFLGCLR register or by an $\overline{XRS}$ reset. No further NMI interrupts pulses are generated until this flag is cleared by the user.
		0	No NMI interrupt generated
		1	NMI interrupt generated

1.13.5.9 C28 NMI Flag Clear (CNMIFLGCLR) Register

NOTE: If hardware is trying to set a bit to "1" while software is trying to clear a bit to "0" on the same cycle, hardware has priority.

Users should clear the pending FAIL flag first and then clear the NMIINT flag.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-66. C28 NMI Flag Clear (CNMIFLGCLR) Register

15							8
Reserved							
R-0:0							

7	6	5	4	3	2	1	0
Reserved	ACIBERR	C28BISTERR	M3BISTERR	C28FLUNCERR	C28RAMUNCERR	CLOCKFAIL	NMIINT
R-0:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-77. C28 NMI Flag Clear (CNMIFLGCLR) Register Field Descriptions

Bit	Field	Value	Description
15-7	Reserved		Reserved
6	ACIBERR		CIB Error NMI Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.
5	C28BISTERR		C28 BIST Error Flag
		0	Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers.
4	M3BISTERR		M3 BIST Error Flag
		0	Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers.
3	C28FLUNCERR		C28 Flash Uncorrectable Error NMI Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.
2	C28RAMUNCERR		C28 RAM Uncorrectable Error NMI Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.
1	CLOCKFAIL		Clock Fail NMI Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.
0	NMIINT		NMI Interrupt Flag Clear
		0	Ignored; always reads back 0.
		1	Clears the corresponding flag bit in the NMIFLG register.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.5.10 C28 NMI Flag Force (CNMIFLGFRC) Register

Figure 1-67. C28 NMI Flag Force (CNMIFLGFRC) Register

15	8
Reserved	
R-0:0	

7	6	5	3	2	1	0
Reserved	ACIBERR	Reserved		C28RAMUNCERR	CLOCKFAIL	Reserved
R-0:0	R/W-0	R-0		R/W-0	R/W-0	R-0:0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-78. C28 NMI Flag Force (CNMIFLGFRC) Register Field Descriptions

Bit	Field	Value	Description
15-7	Reserved		Reserved
6	ACIBERR		CIB Error NMI Flag Force
		0	Ignored; always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Clears the corresponding flag bit in the NMIFLG register.
5-3	Reserved		Reserved
2	C28RAMUNCERR		C28 RAM Uncorrectable Error NMI Flag Force
		0	Ignored; always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Clears the corresponding flag bit in the NMIFLG register.
1	CLOCKFAIL		Clock Fail NMI Flag Force
		0	Ignored; always reads back 0. This can be used as a means to test the NMI mechanisms.
		1	Clears the corresponding flag bit in the NMIFLG register.
0	Reserved		Reserved

1.13.5.11 C28 NMI Watchdog Counter (CNMIWDCNT) Register

Figure 1-68. C28 NMI Watchdog Counter (CNMIWDCNT) Register

15	0
NMIWDCNT	
R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-79. C28 NMI Watchdog Counter (CNMIWDCNT) Register Field Descriptions

Bit	Field	Value	Description
15-0	NMIWDCNT		NMI Watchdog Counter This 16-bit incremental counter will start incrementing whenever any one of the enabled "FAIL" flags are set. If the counter reaches the period value, an $\overline{\text{NMIRS}}$ signal is fired which will then reset the C28 CPU and sub-system. See Section 1.3 for more details on the reseT behavior. The counter will reset to zero when it reaches the period value and will then restart counting if any of the enabled "NMI" flags are set. If no enabled "NMI" flag is set, then the counter will reset to zero and remain at zero until an enabled "NMI" flag is set.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.5.12 C28 NMI Watchdog Period (CNMIWDPRD) Register

Figure 1-69. C28 NMI Watchdog Period (CNMIWDPRD) Register

15	0
NMIWDPRD	
R/W-0xFFFF	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-80. C28 NMI Watchdog Period (CNMIWDPRD) Register Field Descriptions

Bit	Field	Value	Description
15-0	NMIWDPRD		NMI Watchdog Period This 16-bit value contains the period value at which a reset is generated when the watchdog counter matches. At reset this value is set at the maximum. The software can decrease the period value at initialization time. Writing a PERIOD value that is smaller then the current counter value will automatically force an $\overline{\text{NMIRS}}$ to the C28 and hence reset the watchdog counter.

1.13.5.13 PIE Interrupt Registers

1.13.5.13.1 PIE, Control (PIECTRL) Register

Figure 1-70. PIE, Control (PIECTRL) Register

15	1	0
PIEVECT		ENPIE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-81. PIE, Control (PIECTRL) Register Field Descriptions

Bits	Field	Value	Description
15-1	PIEVECT		These bits indicate the address within the PIE vector table from which the vector was fetched. The least significant bit of the address is ignored and only bits 1 to 15 of the address is shown. You can read the vector value to determine which interrupt generated the vector fetch. **For Example:** If PIECTRL = 0x0D27 then the vector from address 0x0D26 (illegal operation) was fetched.
0	ENPIE		Enable vector fetching from PIE vector table. **Note:** The reset vector is never fetched from the PIE, even when it is enabled. This vector is always fetched from boot ROM.
		0	If this bit is set to 0, the PIE block is disabled and vectors are fetched from the CPU vector table in boot ROM. All PIE block registers (PIEACK, PIEIFR, PIEIER) can be accessed even when the PIE block is disabled.
		1	When ENPIE is set to 1, all vectors, except for reset, are fetched from the PIE vector table. The reset vector is always fetched from the boot ROM.

1.13.5.13.2 PIE, Acknowledge (PIEACK) Register

Figure 1-71. PIE, Acknowledge (PIEACK) Register

15	12	11	0
Reserved		PIEACK	
R-0		R/W1C-1	

LEGEND: R/W1C = Read/Write 1 to clear; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-82. PIE, Acknowledge (PIEACK) Register Field Descriptions

Bits	Field	Value	Description
15-12	Reserved		Reserved
11-0	PIEACK		Each bit in PIEACK refers to a specific PIE group. Bit 0 refers to interrupts in PIE group 1 that are MUXed into $\overline{\text{INT1}}$ up to Bit 11, which refers to PIE group 12 which is MUXed into CPU $\overline{\text{INT12}}$.
		bit x = 0 (1)	If a bit reads as a 0, it indicates that the PIE can send an interrupt from the respective group to the CPU.
			Writes of 0 are ignored.
		bit x = 1	Reading a 1 indicates if an interrupt from the respective group has been sent to the CPU and all other interrupts from the group are currently blocked.
			Writing a 1 to the respective interrupt bit clears the bit and enables the PIE block to drive a pulse into the CPU interrupt input if an interrupt is pending for that group.

(1) bit x = PIEACK bit 0 - PIEACK bit 11. Bit 0 refers to CPU $\overline{\text{INT1}}$ up to Bit 11, which refers to CPU $\overline{\text{INT12}}$

1.13.5.13.3 PIE Interrupt Enable Registers

There are twelve PIEIER registers, one for each CPU interrupt used by the PIE module (INT1-INT12).

Figure 1-72. PIE, INTx Group Enable Register (PIEIERx) (x = 1 to 12)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-83. PIE, INTx Group Enable Register (PIEIERx) (x = 1 to 12) Field Descriptions

Bits	Field	Description
15-8	Reserved	Reserved
7	INTx.8	These register bits individually enable an interrupt within a group and behave very much like the core interrupt enable register. Setting a bit to 1 enables the servicing of the respective interrupt. Setting a bit to 0 disables the servicing of the interrupt. x = 1 to 12. INTx means CPU INT1 to INT12
6	INTx.7	
5	INTx.6	
4	INTx.5	
3	INTx.4	
2	INTx.3	
1	INTx.2	
0	INTx.1	

NOTE: Care must be taken when clearing PIEIER bits during normal operation. See Section Section 1.5.4.3.2 for the proper procedure for handling these bits.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.5.13.4 PIE Interrupt Flag Registers

There are twelve PIEIFR registers, one for each CPU interrupt used by the PIE module (INT1-INT12).

Figure 1-73. PIE, INTx Group Flag Register (PIEIFRx) (x = 1 to 12)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-84. PIE, INTx Group Flag Register (PIEIFRx) (x = 1 to 12) Field Descriptions

Bits	Field	Description
15-8	Reserved	Reserved
7	INTx.8	These register bits indicate whether an interrupt is currently active. They behave very much like the CPU interrupt flag register. When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit. This register can also be read to determine which interrupts are active or pending. x = 1 to 12. INTx means CPU INT1 to INT12 The PIEIFR register bit is cleared during the interrupt vector fetch portion of the interrupt processing. Hardware has priority over CPU accesses to the PIEIFR registers.
6	INTx.7	
5	INTx.6	
4	INTx.5	
3	INTx.4	
2	INTx.3	
1	INTx.2	
0	INTx.1	

NOTE: Never clear a PIEIFR bit. An interrupt may be lost during the read-modify-write operation. See Section Section 1.5.4.3.1 for a method to clear flagged interrupts.

1.13.5.13.5 CPU Interrupt Flag Register (IFR)

The CPU interrupt flag register (IFR), is a 16-bit, CPU register and is used to identify and clear pending interrupts. The IFR contains flag bits for all the maskable interrupts at the CPU level (INT1-INT14, DLOGINT and RTOSINT). When the PIE is enabled, the PIE module multiplexes interrupt sources for INT1-INT12.

When a maskable interrupt is requested, the flag bit in the corresponding peripheral control register is set to 1. If the corresponding mask bit is also 1, the interrupt request is sent to the CPU, setting the corresponding flag in the IFR. This indicates that the interrupt is pending or waiting for acknowledgment.

To identify pending interrupts, use the PUSH IFR instruction and then test the value on the stack. Use the OR IFR instruction to set IFR bits and use the AND IFR instruction to manually clear pending interrupts. All pending interrupts are cleared with the AND IFR #0 instruction or by a hardware reset.

The following events also clear an IFR flag:

- The CPU acknowledges the interrupt.
- The 28x device is reset.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE:

1. To clear a CPU IFR bit, you must write a zero to it, not a one.
2. When a maskable interrupt is acknowledged, only the IFR bit is cleared automatically. The flag bit in the corresponding peripheral control register is not cleared. If an application requires that the control register flag be cleared, the bit must be cleared by software.
3. When an interrupt is requested by an INTR instruction and the corresponding IFR bit is set, the CPU does not clear the bit automatically. If an application requires that the IFR bit be cleared, the bit must be cleared by software.
4. IMR and IFR registers pertain to core-level interrupts. All peripherals have their own interrupt mask and flag bits in their respective control/configuration registers. Note that several peripheral interrupts are grouped under one core-level interrupt.

Figure 1-74. CPU Interrupt Flag Register (IFR)

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-85. CPU Interrupt Flag Register Field Descriptions (IFR)

Bits	Field	Value	Description
15	RTOSINT		Real-time operating system flag. RTOSINT is the flag for RTOS interrupts.
		0	No RTOS interrupt is pending
		1	At least one RTOS interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
14	DLOGINT		Data logging interrupt fag. DLOGINT is the flag for data logging interrupts.
		0	No DLOGINT is pending
		1	At least one DLOGINT interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
13	INT14		Interrupt 14 flag. INT14 is the flag for interrupts connected to CPU interrupt level INT14.
		0	No INT14 interrupt is pending
		1	At least one INT14 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
12	INT13		Interrupt 13 flag. INT13 is the flag for interrupts connected to CPU interrupt level INT13I.
		0	No INT13 interrupt is pending
		1	At least one INT13 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
11	INT12		Interrupt 12 flag. INT12 is the flag for interrupts connected to CPU interrupt level INT12.
		0	No INT12 interrupt is pending
		1	At least one INT12 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
10	INT11		Interrupt 11 flag. INT11 is the flag for interrupts connected to CPU interrupt level INT11.
		0	No INT11 interrupt is pending
		1	At least one INT11 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
9	INT10		Interrupt 10 flag. INT10 is the flag for interrupts connected to CPU interrupt level INT10.
		0	No INT10 interrupt is pending
		1	At least one INT6 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-85. CPU Interrupt Flag Register Field Descriptions (IFR) (continued)

Bits	Field	Value	Description
8	INT9		Interrupt 9 flag. INT9 is the flag for interrupts connected to CPU interrupt level INT6.
		0	No INT9 interrupt is pending
		1	At least one INT9 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
7	INT8		Interrupt 8 flag. INT8 is the flag for interrupts connected to CPU interrupt level INT6.
		0	No INT8 interrupt is pending
		1	At least one INT8 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
6	INT7		Interrupt 7 flag. INT7 is the flag for interrupts connected to CPU interrupt level INT7.
		0	No INT7 interrupt is pending
		1	At least one INT7 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
5	INT6		Interrupt 6 flag. INT6 is the flag for interrupts connected to CPU interrupt level INT6.
		0	No INT6 interrupt is pending
		1	At least one INT6 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
4	INT5		Interrupt 5 flag. INT5 is the flag for interrupts connected to CPU interrupt level INT5.
		0	No INT5 interrupt is pending
		1	At least one INT5 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
3	INT4		Interrupt 4 flag. INT4 is the flag for interrupts connected to CPU interrupt level INT4.
		0	No INT4 interrupt is pending
		1	At least one INT4 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
2	INT3		Interrupt 3 flag. INT3 is the flag for interrupts connected to CPU interrupt level INT3.
		0	No INT3 interrupt is pending
		1	At least one INT3 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
1	INT2		Interrupt 2 flag. INT2 is the flag for interrupts connected to CPU interrupt level INT2.
		0	No INT2 interrupt is pending
		1	At least one INT2 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
0	INT1		Interrupt 1 flag. INT1 is the flag for interrupts connected to CPU interrupt level INT1.
		0	No INT1 interrupt is pending
		1	At least one INT1 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request

1.13.5.13.6 CPU Interrupt Enable Register (IER)

The IER is a 16-bit CPU register. The IER contains enable bits for all the maskable CPU interrupt levels (INT1-INT14, RTOSINT and DLOGINT). Neither NMI nor XRS is included in the IER; thus, IER has no effect on these interrupts.

You can read the IER to identify enabled or disabled interrupt levels, and you can write to the IER to enable or disable interrupt levels. To enable an interrupt level, set its corresponding IER bit to one using the OR IER instruction. To disable an interrupt level, set its corresponding IER bit to zero using the AND IER instruction. When an interrupt is disabled, it is not acknowledged, regardless of the value of the INTM bit. When an interrupt is enabled, it is acknowledged if the corresponding IFR bit is one and the INTM bit is zero.

When using the OR IER and AND IER instructions to modify IER bits make sure they do not modify the state of bit 15 (RTOSINT) unless a real-time operating system is present.

Copyright © 2012–2019, Texas Instruments Incorporated

When a hardware interrupt is serviced or an INTR instruction is executed, the corresponding IER bit is cleared automatically. When an interrupt is requested by the TRAP instruction the IER bit is not cleared automatically. In the case of the TRAP instruction if the bit needs to be cleared it must be done by the interrupt service routine.

At reset, all the IER bits are cleared to 0, disabling all maskable CPU level interrupts.

The IER register is shown in Figure 1-75, and descriptions of the bits follow the figure.

Figure 1-75. CPU Interrupt Enable Register (IER)

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-86. CPU Interrupt Enable Register (IER) Field Descriptions

Bits	Field	Value	Description
15	RTOSINT		Real-time operating system interrupt enable. RTOSINT enables or disables the CPU RTOS interrupt.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
14	DLOGINT		Data logging interrupt enable. DLOGINT enables or disables the CPU data logging interrupt.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
13	INT14		Interrupt 14 enable. INT14 enables or disables CPU interrupt level INT14.
		0	Level INT14 is disabled
		1	Level INT14 is enabled
12	INT13		Interrupt 13 enable. INT13 enables or disables CPU interrupt level INT13.
		0	Level INT13 is disabled
		1	Level INT13 is enabled
11	INT12		Interrupt 12 enable. INT12 enables or disables CPU interrupt level INT12.
		0	Level INT12 is disabled
		1	Level INT12 is enabled
10	INT11		Interrupt 11 enable. INT11 enables or disables CPU interrupt level INT11.
		0	Level INT11 is disabled
		1	Level INT11 is enabled
9	INT10		Interrupt 10 enable. INT10 enables or disables CPU interrupt level INT10.
		0	Level INT10 is disabled
		1	Level INT10 is enabled
8	INT9		Interrupt 9 enable. INT9 enables or disables CPU interrupt level INT9.
		0	Level INT9 is disabled
		1	Level INT9 is enabled
7	INT8		Interrupt 8 enable. INT8 enables or disables CPU interrupt level INT8.
		0	Level INT8 is disabled
		1	Level INT8 is enabled
6	INT7		Interrupt 7 enable. INT7 enables or disables CPU interrupt level INT7.
		0	Level INT7 is disabled
		1	Level INT7 is enabled

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-86. CPU Interrupt Enable Register (IER) Field Descriptions (continued)

Bits	Field	Value	Description
5	INT6		Interrupt 6 enable. INT6 enables or disables CPU interrupt level INT6.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
4	INT5		Interrupt 5 enable.INT5 enables or disables CPU interrupt level INT5.
		0	Level INT5 is disabled
		1	Level INT5 is enabled
3	INT4		Interrupt 4 enable.INT4 enables or disables CPU interrupt level INT4.
		0	Level INT4 is disabled
		1	Level INT4 is enabled
2	INT3		Interrupt 3 enable.INT3 enables or disables CPU interrupt level INT3.
		0	Level INT3 is disabled
		1	Level INT3 is enabled
1	INT2		Interrupt 2 enable.INT2 enables or disables CPU interrupt level INT2.
		0	Level INT2 is disabled
		1	Level INT2 is enabled
0	INT1		Interrupt 1 enable.INT1 enables or disables CPU interrupt level INT1.
		0	Level INT1 is disabled
		1	Level INT1 is enabled

1.13.5.13.7 Debug Interrupt Enable Register (DBGIER)

The Debug Interrupt Enable Register (DBGIER) is used only when the CPU is halted in real-time emulation mode. An interrupt enabled in the DBGIER is defined as a time-critical interrupt. When the CPU is halted in real-time mode, the only interrupts that are serviced are time-critical interrupts that are also enabled in the IER. If the CPU is running in real-time emulation mode, the standard interrupt-handling process is used and the DBGIER is ignored.

As with the IER, you can read the DBGIER to identify enabled or disabled interrupts and write to the DBGIER to enable or disable interrupts. To enable an interrupt, set its corresponding bit to 1. To disable an interrupt, set its corresponding bit to 0. Use the PUSH DBGIER instruction to read from the DBGIER and POP DBGIER to write to the DBGIER register. At reset, all the DBGIER bits are set to 0.

Figure 1-76. Debug Interrupt Enable Register (DBGIER)

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-87. Debug Interrupt Enable Register (DBGIER) Field Descriptions

Bits	Field	Value	Description
15	RTOSINT		Real-time operating system interrupt enable. RTOSINT enables or disables the CPU RTOS interrupt.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
14	DLOGINT	.	Data logging interrupt enable. DLOGINT enables or disables the CPU data logging interrupt
		0	Level INT6 is disabled
		1	Level INT6 is enabled

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-87. Debug Interrupt Enable Register (DBGIER) Field Descriptions (continued)

Bits	Field	Value	Description
13	INT14	.	Interrupt 14 enable. INT14 enables or disables CPU interrupt level INT14
		0	Level INT14 is disabled
		1	Level INT14 is enabled
12	INT13		Interrupt 13 enable. INT13 enables or disables CPU interrupt level INT13.
		0	Level INT13 is disabled
		1	Level INT13 is enabled
11	INT12		Interrupt 12 enable. INT12 enables or disables CPU interrupt level INT12.
		0	Level INT12 is disabled
		1	Level INT12 is enabled
10	INT11		Interrupt 11 enable. INT11 enables or disables CPU interrupt level INT11.
		0	Level INT11 is disabled
		1	Level INT11 is enabled
9	INT10		Interrupt 10 enable. INT10 enables or disables CPU interrupt level INT10.
		0	Level INT10 is disabled
		1	Level INT10 is enabled
8	INT9		Interrupt 9 enable. INT9 enables or disables CPU interrupt level INT9.
		0	Level INT9 is disabled
		1	Level INT9 is enabled
7	INT8		Interrupt 8 enable. INT8 enables or disables CPU interrupt level INT8.
		0	Level INT8 is disabled
		1	Level INT8 is enabled
6	INT7		Interrupt 7 enable. INT7 enables or disables CPU interrupt level INT77.
		0	Level INT7 is disabled
		1	Level INT7 is enabled
5	INT6		Interrupt 6 enable. INT6 enables or disables CPU interrupt level INT6.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
4	INT5		Interrupt 5 enable.INT5 enables or disables CPU interrupt level INT5.
		0	Level INT5 is disabled
		1	Level INT5 is enabled
3	INT4		Interrupt 4 enable.INT4 enables or disables CPU interrupt level INT4.
		0	Level INT4 is disabled
		1	Level INT4 is enabled
2	INT3		Interrupt 3 enable.INT3 enables or disables CPU interrupt level INT3.
		0	Level INT3 is disabled
		1	Level INT3 is enabled
1	INT2		Interrupt 2 enable.INT2 enables or disables CPU interrupt level INT2.
		0	Level INT2 is disabled
		1	Level INT2 is enabled
0	INT1		Interrupt 1 enable.INT1 enables or disables CPU interrupt level INT1.
		0	Level INT1 is disabled
		1	Level INT1 is enabled

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.5.14 C28 External Interrupt 1 Configuration Register (XINT1CR)

Figure 1-77. C28 External Interrupt 1 Configuration Register (XINT1CR)

15	4	3	2	1	0
Reserved		POLARITY		Rsvd	ENABLE
R-0		R/W-0:0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-88. C28 External Interrupt 1 Configuration Register (XINT1CR) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3-2	POLARITY		C28 XINT1 Polarity
		0,0	Interrupt is selected as negative edge triggered.
		0,1	Interrupt is selected as positive edge triggered.
		1,0	Interrupt is selected as negative edge triggered.
		1,1	Interrupt is selected as positive or negative edge triggered.
1	Reserved		Reserved
0	ENABLE		C28 XINT1 Enable
		0	Interrupt disabled
		1	Interrupt enabled

1.13.5.15 C28 External Interrupt 2 Configuration Register (XINT2CR)

Figure 1-78. C28 External Interrupt 2 Configuration Register (XINT2CR)

15	4	3	2	1	0
Reserved		POLARITY		Rsvd	ENABLE
R-0		R/W-0:0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-89. C28 External Interrupt 2 Configuration Register (XINT2CR) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3-2	POLARITY		C28 XINT2 Polarity
		0,0	Interrupt is selected as negative edge triggered.
		0,1	Interrupt is selected as positive edge triggered.
		1,0	Interrupt is selected as negative edge triggered.
		1,1	Interrupt is selected as positive or negative edge triggered.
1	Reserved		Reserved
0	ENABLE		C28 XINT2 Enable
		0	Interrupt disabled
		1	Interrupt enabled

1.13.5.16 C28 External Interrupt 3 Configuration Register (XINT3CR)

Figure 1-79. C28 External Interrupt 3 Configuration Register (XINT3CR)

15	4	3	2	1	0
Reserved		POLARITY		Rsvd	ENABLE
R-0		R/W-0:0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-90. C28 External Interrupt 3 Configuration Register (XINT3CR) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3-2	POLARITY		C28 XINT3 Polarity
		0,0	Interrupt is selected as negative edge triggered.
		0,1	Interrupt is selected as positive edge triggered.
		1,0	Interrupt is selected as negative edge triggered.
		1,1	Interrupt is selected as positive or negative edge triggered.
1	Reserved		Reserved
0	ENABLE		C28 XINT3 Enable
		0	Interrupt disabled
		1	Interrupt enabled

1.13.5.17 C28 External Interrupt 1 Counter Register (XINT1CTR)

Figure 1-80. C28 External Interrupt 1 Counter Register (XINT1CTR)

15	0
INTCTR	
R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-91. C28 External Interrupt 1 Counter Register (XINT1CTR) Field Descriptions

Bit	Field	Value	Description
15-0	INTCTR		C28 XINT1 Counter This is a free running 16-bit up-counter that is clocked at the C28SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. The counter must only be reset by the selected POLARITY edge as selected in the respective interrupt control register. When the interrupt is disabled, the counter will stop. The counter is a free-running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by the C28 SYSRSN reset.

1.13.5.18 C28 External Interrupt 2 Counter Register (XINT2CTR)

Figure 1-81. C28 External Interrupt 2 Counter Register (XINT2CTR)

15	0
INTCTR	
R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-92. C28 External Interrupt 2 Counter Register (XINT2CTR) Field Descriptions

Bit	Field	Value	Description
15-0	INTCTR		C28 XINT2 Counter This is a free running 16-bit up-counter that is clocked at the C28SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. The counter must only be reset by the selected POLARITY edge as selected in the respective interrupt control register. When the interrupt is disabled, the counter will stop. The counter is a free-running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by the C28 SYSRSN reset.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.5.19 C28 External Interrupt 3 Counter Register (XINT3CTR)

Figure 1-82. C28 External Interrupt 3 Counter Register (XINT3CTR)

15	0
INTCTR	
R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-93. C28 External Interrupt 3 Counter Register (XINT3CTR) Field Descriptions

Bit	Field	Value	Description
15-0	INTCTR		C28 XINT3 Counter This is a free running 16-bit up-counter that is clocked at the C28SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. The counter must only be reset by the selected POLARITY edge as selected in the respective interrupt control register. When the interrupt is disabled, the counter will stop. The counter is a free-running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by the C28 SYSRSN reset.

1.13.5.20 System PLL Configuration (SYSPLLCTL) Register

Figure 1-83. System PLL Configuration (SYSPLLCTL) Register

31	16
Reserved	
R-0:0	

15	8
Reserved	
R-0:0 R-0	

7	2	1	0
Reserved		SPLLCLKEN	SPLLEN
R-0:0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-94. System PLL Configuration (SYSPLLCTL) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	SPLLCLKEN		System PLL Clock Enable System PLL bypassed or included in the PLLSYSCLK path. This bit decides if the system PLL is bypassed when PLLSYSCLK is generated.
		0	System PLL is bypassed; clock to the system is a direct feed from X1.
		1	System PLL is on the clock path to PLLSYSCLK; PLLSYSCLK is the PLL multiplied clock.
0	SPLLEN		System PLL Enable This bit decides if the system PLL is enabled or not.
		0	System PLL is powered off; clock to the system is a direct feed from X1.
		1	System PLL is enabled and clock to the system will depend on SYSPLLMULT and SYSPLLCLKEN register bit configuration.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.5.21 Control Subsystem Clock Disable (CCLKOFF) Register

Figure 1-84. Control Subsystem Clock Disable (CCLKOFF) Register

31	1	0
Reserved		C28CLKINDIS
R-0:0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-95. Control Subsystem Clock Disable (CCLKOFF) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	C28CLKINDIS		C28 CPU CLKIN Disable This bit decides if the C28 CPU gets a clock or not.
		0	C28 CPU CLKIN is on and is the same frequency as PLLSYSCLK.
		1	C28 CPU CLKIN is turned off.

1.13.6 Safety Control Registers

1.13.6.1 M3 Configuration Write Allow (MWRALLOW) Register

Figure 1-85. M3 Configuration Write Allow (MWRALLOW) Register

31	0
ALLOW	
R/W-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-96. M3 Configuration Write Allow (MWRALLOW) Register Field Descriptions

Bit	Field	Value	Description
31-0	ALLOW		M3 Write Allow Bits These bits when set to "0xA5A5A5A5" enable the write to all other protected mode register writes on the M3 subsystem. This register can be written to only by the M3 in privilege mode.
		0xA5A5A5A5	Protected register writes allowed
		Any other value	Protected register writes disabled

1.13.6.2 M3 Configuration Lock (MLOCK) Register

Figure 1-86. M3 Configuration Lock (MLOCK) Register

31	1	0
Reserved		MSxMSELLOCK
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-97. M3 Configuration Lock (MLOCK) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	MSxMSELLOCK		Lock Writes to MSxMSEL Register This is a write once only register bit. It prevents further writes to the MSxMSEL register and overrides any other protection mechanism. Reading the bit gives the lock state.Lock mechanism can only be cleared by shared resource reset.
		0	Ignored
		1	Lock the register

1.13.6.3 Missing Clock Status (MCLKSTS) Register

Figure 1-87. Missing Clock Status (MCLKSTS) Register

31	17	16
Reserved		MCLKFLG
R-0:0		R-0

15	8	7	0
Reserved		REFCLKCNT	
R-0:0		R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-98. Missing Clock Status (MCLKSTS) Register Field Descriptions

Bit	Field	Value	Description
31-17	Reserved		Reserved
16	MCLKFLG		Missing Clock Status Flag Clock missing condition detection status flag bit.
		0	Reference clock is not missing, clock source is X1
		1	Reference clock is missing, clock source is internal oscillator
15-8	Reserved		Reserved
7-0	REFCLKCNT		Reference Clock Count Read only value of the reference clock counter. **Note:** This is a free-running counter that continues to run even after clock glitches causing a missing clock condition.

1.13.6.4 Missing Clock Force (MCLKFRCCLR) Register

Figure 1-88. Missing Clock Force (MCLKFRCCLR) Register

31	17	16
Reserved		MCLKCLR
R-0:0		R/W-0

15	1	0
Reserved		REFCLKOFF
R-0:0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-99. Missing Clock Force (MCLKFRCCLR) Register Field Descriptions

Bit	Field	Value	Description
31-17	Reserved		Reserved
16	MCLKCLR		Missing Clock Status Flag Write "1" to this bit to clear the MCLKFLG bit in the MCLKSTAT register. Always reads "0". **Note:** Hardware should switch the clock source to the oscillator reference clock input only when the missing clock condition is cleared by software.
		0	MCLKSTS remains in the same state.
		1	Clear MCLKSTS to "0".
15-1	Reserved		Reserved
0	REFCLKOFF		Reference Clock Off Setting this bit will switch off reference clocks to the missing clock detection circuit, thereby forcing a clock missing condition in the design. Used for software development and test.
		0	Enable reference clock input to the missing clock logic.
		1	Force reference clock off to the missing clock logic.

1.13.6.5 Missing Clock Enable (MCLKEN) Register

Figure 1-89. Missing Clock Enable (MCLKEN) Register

31	16
Reserved	
R-0:0	

15	9	8
Reserved		MCLKNMIEN
R-0:0		R/W-1

7	0
Reserved	
R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-100. Missing Clock Enable (MCLKEN) Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	MCLKNMIEN		Missing Clock NMI Enable When set, the missing clock logic will generate an NMI input to the M3 and C28 CPUs when a clock missing condition is detected. This bit can be used along with REFCLKOFF bit to test software and debug.
		0	Missing clock detection will not generate an NMI to the M3 and C28 CPUs.
		1	Missing clock NMI will be sent to the M3 and C28 CPUs.
7-0	Reserved		Reserved

1.13.6.6 Missing Clock Reference Limit (MCLKLIMIT) Register

Figure 1-90. Missing Clock Reference Limit (MCLKLIMIT) Register

31	16
Reserved	
R-0:0	

15	8	7	0
REFCLKHILIMIT		REFCLKLOLIMIT	
R/W-0x7A		R/W-0x2	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-101. Missing Clock Reference Limit (MCLKLIMIT) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-8	REFCLKHILIMIT		Reference Clock High Limit Contains the higher limit for the reference clock counter.
7-0	REFCLKLOLIMIT		Reference Clock Low Limit Contains the lower limit for the reference clock counter.

NOTE: Due to the asynchronous nature of the missing clock detect block, the MCLKDETEN = 0 write will take 3 cycles (2 cycles best case) for the 10MHZ clock to generate the synchronized reset for the internal counters. Hence if MCLKDETEN = 0 and REFCLKHILIMIT/REFCLKLOLIMIT values are set back to back, then MCLKDETEN = 0, will not propagate to the counters. As a result, the new REFCLKHILIMIT/REFCLKLOLIMIT values will take effect and may generate a missing_clock condition.

This can be avoided by adding three empty cycles between setting MCLKDETEN = 0 and setting the REFCLKHILIMIT and REFCLKLOLIMIT values.

1.13.6.7 C28 USER_SWREG1 Register

This register is used by the PIE mismatch handler in C-Boot ROM as mentioned in the safety features section of this chapter. The user should initialize this register with the lower 16 bits of the application PIE mismatch handler address. If the user doesn't wish to install a PIE mismatch handler, this register is to be left at its reset value.

Figure 1-91. C28 USER_SWREG1 Register

15	0
SWREG1	
R/W-0xFFFF	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-102. C28 USER_SWREG1 Register Field Descriptions

Bit	Field	Value	Description
15-0	SWREG1		General purpose register for C28 software use.

1.13.6.8 C28_USER_SWREG2 Register

This register is used by the PIE mismatch handler in C-Boot ROM as mentioned in the safety features section of this chapter. The user should initialize the lower 8 bits of this register with the upper 8 bits of the application PIE mismatch handler address (note that on the C28x addresses are limited to 22 bits). If the user does not wish to install a PIE mismatch handler then this register is to be left at its reset value.

Figure 1-92. C28_USER_SWREG2 Register

15	14 9	8	7 0
Reserved	SWREG2	Reserved	SWREG2
R-0	R/W-0:0	R-0	R/W-0:0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-103. C28_USER_SWREG2 Register Field Descriptions

Bit	Field	Value	Description
15	Reserved		Reserved
14-9	SWREG2		General purpose register for C28 software use.
		0	
		1	
8	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

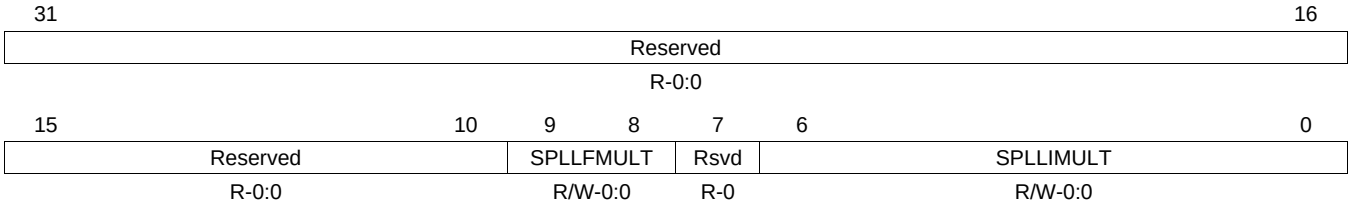
Table 1-103. C28_USER_SWREG2 Register Field Descriptions (continued)

Bit	Field	Value	Description
7-0	SWREG2		General purpose register for C28 software use.
		0	
		1	

1.13.7 Clocking Control Registers

1.13.7.1 System PLL Multiplier (SYSPLLMULT) Register

Figure 1-93. System PLL Multiplier (SYSPLLMULT) Register



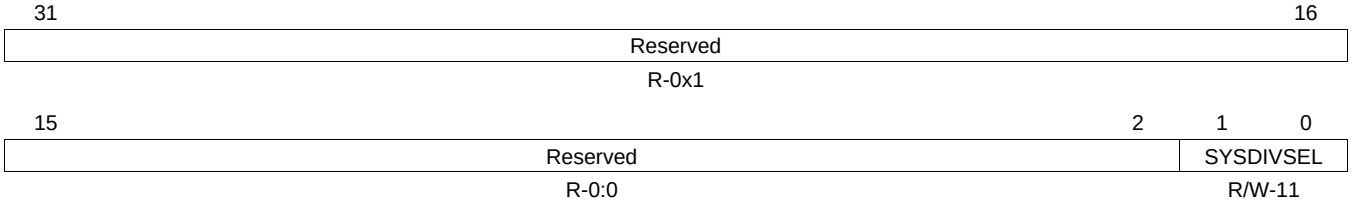
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-104. System PLL Multiplier (SYSPLLMULT) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Reserved
9-8	SPLLFMULT		System PLL Fractional Mutliplier
		00	Fractional multiplier = 0
		01	Fractional multiplier = 0.25
		10	Fractional multiplier = 0.5
		11	Fractional multiplier = 0.75
7	Reserved		Reserved
6-0	SPLLIMULT		System PLL Integer Multiplier
		0000000	Integer multiplier = 1
		0000001	Integer multiplier = 1
		0000010	Integer multiplier = 2
		0000011	Integer multiplier = 3
		…	…
		1111111	Integer multipler = 127

1.13.7.2 System Clock Divider (SYSDIVSEL) Register

Figure 1-94. System Clock Divider (SYSDIVSEL) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-105. System Clock Divider (SYSDIVSEL) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1-0	SYSDIVSEL		System Clock Divide Select This bit selects between /8, /4, /2 and /1 for PLLSYSCLK (CLKIN to the C28 CPU as well as input clock to the M3 subsystem and CIB clock pre-scalers). This clock is referred to as the C28 SS clock. The configuration of the SYSDIVSEL bits is as follows.
		00	Select C28 CLKIN Divide By 1 of PLLSYSCLK
		01	Select C28 CLKIN Divide By 2 of PLLSYSCLK
		10	Select C28 CLKIN Divide By 4 of PLLSYSCLK
		11	Select C28 CLKIN Divide By 8 (at reset) of PLLSYSCLK

1.13.7.3 System PLL Lock Status (SYSPLLSTS) Register

Figure 1-95. System PLL Lock Status (SYSPLLSTS) Register

31	2	1	0
Reserved		SPLLSLIPS	SYSPLLLOCKS
R-0:0		R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-106. System PLL Lock Status (SYSPLLSTS) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	SPLLSLIPS		System PLL Out-of-Range Status This bit indicates whether the PLL is out of lock range. **Note:** If a PLL out-of-lock condition is detected then an interrupt is generated to the NVIC; software can decide to relock the PLL or switch to PLL bypass mode in the interrupt handler.
		0	System PLL is not out of lock
		1	System PLL is out of lock
0	SYSPLLLOCKS		System PLL Lock Status This bit indicates whether the PLL is locked or not.
		0	System PLL is not locked
		1	System PLL is locked

1.13.7.4 Master Subsystem Clock Divider (M3SSDIVSEL) Register

Figure 1-96. Master Subsystem Clock Divider (M3SSDIVSEL) Register

31	2	1	0
Reserved		M3SSDIVSEL	
R-0:0		R/W-10	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-107. Master Subsystem Clock Divider (M3SSDIVSEL) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1-0	M3SSDIVSEL		M3 Subsystem Clock Divide This bit selects between /4, /2, and /1 for the M3 sub-system clock. The C28 CLKIN clock is divided by the below ratios to generate the M3 SS clock. The configuration of the M3SSDIVSEL bits is as follows.
		00	Select M3SS clock divide By 1 of PLLYSCLK clock
		01	Select M3SS clock divide By 2 of PLLYSCLK clock
		10	Select M3SS clock divide By 4 of PLLSYSCLK clock
		11	Reserved

1.13.7.5 XPLL CLKOUT Control (XPLLCLKCFG) Register

Figure 1-97. XPLL CLKOUT Control (XPLLCLKCFG) Register

31	2	1	0
Reserved		XPLLCLKOUTDIV	
R-0:0		R/W-11	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-108. XPLL CLKOUT Control (XPLLCLKCFG) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1-0	XPLLCLKOUTDIV		XPLLCLKOUT Divide Ratio This bit selects a clock divide ratio for the XPLLCLKOUT clock. The configuration of the XPLLCLKOUTDIV bits is as follows.
			For Rev0 Devices only
		00	XPLLCLKOUT is off
		01	Select XPLLCLKOUT = PLLSYSCLKC28
		10	Select XPLLCLKOUT = PLLSYSCLK /2
		11	Select XPLLCLKOUT = PLLSYSCLK /4 (default)
			For RevA Devices and newer
		00	XPLLCLKOUT is off
		01	Select XPLLCLKOUT = C28 SYSCLK/4
		10	Select XPLLCLKOUT = M3 SSCLK/4
		11	Select XPLLCLKOUT = PLLSYSCLK/4 (default)

1.13.7.6 USB PLL Configuration (UPLLCTL) Register

Figure 1-98. USB PLL Configuration (UPLLCTL) Register

31	3	2	1	0
Reserved		UPLLCLKEN	UPLLEN	
R-0:0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-109. USB PLL Configuration (UPLLCTL) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	UPLLCLKEN		USB PLL Clock Enable USB PLL bypassed or included in the USB clock path. This bit decides if the USB PLL is bypassed to supply the 60-MHz clock to the USB **Note:** The PLL bypass option can be used only with GPIO_XCLKIN as a clock source to the USB because the oscillators will support only up to a 20 MHz input clock
		0	USB PLL is bypassed; clock to the USB is direct feed from GPIO_XCLKIN.
		1	USB PLL is on the clock path to USBCLK and it is the PLL multiplied clock.
1	UPLLEN		USB PLL Enable PLL enabled or disabled. This bit decides if the USB PLL is enabled or not.
		0	USB PLL is powered off and the clock to the USB is a direct feed from the input clock source as decided by the USBPLLCLKSRCSEL bit.
		1	USB PLL is enabled and the clock to the USB will depend on the USBPLLCR register configuration.
0	UPLLCLKSRCSEL		USB PLL Clock Source Select This bit selects the source for the USB PLL input clock. On $\overline{XRS}$ low and after $\overline{XRS}$ goes high, X1 is selected as clock source to the USB PLL by default. The user would need to select X1 or GPIO_XCLKIN as the clock source during their initialization process. Whenever the user changes the clock source, using these bits, the USBPLLCR register will be automatically forced to zero. This prevents potential PLL overshoot. The user will then have to write to the USBPLLCR register to configure the appropriate divisor ratio.
		0	X1 clock source is selected as clock to the USB PLL
		1	GPIO_XCLKIN is selected as clock to the USB PLL

1.13.7.7 USB PLL Multiplier (UPLLMULT) Register

NOTE: Application must take care to program the USBPLLCR register in such a way that the output clock is always 60 MHz. There is no clock divider after the USB PLL OUT.

Figure 1-99. USB PLL Multiplier (UPLLMULT) Register

31	16
Reserved	
R-0:0	

15	10	9	8	7	6	5	0
Reserved		UPLLFMULT		Reserved		UPLLIMULT	
R-0:0		R/W-0:0		R-0:0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-110. USB PLL Multiplier (UPLLMULT) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Reserved
9-8	UPLLFMULT		USB PLL Fractional Multiplier
		00	Fractional multiplier = 0
		01	Fractional multiplier = 0.25
		10	Fractional multiplier = 0.5
		11	Fractional multiplier = 0.75
7-6	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-110. USB PLL Multiplier (UPLLMULT) Register Field Descriptions (continued)

Bit	Field	Value	Description
5-0	UPLLIMULT		USB PLL Integer Multiplier
		000000	Integer multiplier = 1
		000001	Integer multiplier = 2
		000010	Integer multiplier = 3
		000011	Integer multiplier = 4
		…	…
		111111	Integer multipler = 63

1.13.7.8 USB PLL Lock Status (UPLLSTS) Register

Figure 1-100. USB PLL Lock Status (UPLLSTS) Register

31	2	1	0
Reserved		UPLLSLIPS	UPLLLOCKS
R-0:0		R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-111. USB PLL Lock Status (UPLLSTS) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	UPLLSLIPS		USB PLL Out-of-Range Status This bit indicates whether the PLL is out of lock range. **Note:** If PLL out-of-lock condition is detected then an interrupt is generated to the NVIC; software can decide to relock the PLL or switch to PLL bypass mode in the interrupt handler.
		0	USB PLL is not out of lock
		1	USB PLL is out of lock
0	UPLLLOCKS		USB PLL Lock Status This bit indicates whether the PLL is locked or not.
		0	USB PLL is not locked
		1	USB PLL is locked

1.13.7.9 Bit Clock Source Selection for CAN0 (CAN0BCLKSEL) Register

Figure 1-101. Bit Clock Source Selection for CAN0 (CAN0BCLKSEL) Register

31	2	1	0
Reserved		BCLKSEL	
R-0:0		00	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-112. Bit Clock Source Selection for CAN0 (CAN0BCLKSEL) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1-0	BCLKSEL		Bit Clock Source Select
		00	CAN0 bit clock = M3 SS_CLK
		01	CAN0 bit clock = OSCCLK
		1x	CAN0 bit clock = GPIO_XCLKIN

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.7.10 Bit Clock Source Selection for CAN1 (CAN1BCLKSEL) Register

Figure 1-102. Bit Clock Source Selection for CAN1 (CAN1BCLKSEL) Register

31	2	1	0
Reserved		BCLKSEL	
R-0:0		00	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-113. Bit Clock Source Selection for CAN1 (CAN1BCLKSEL) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1-0	BCLKSEL		Bit Clock Source Select
		00	CAN1 bit clock = M3 SS_CLK
		01	CAN1 bit clock = OSCCLK
		1x	CAN1 bit clock = GPIO_XCLKIN

1.13.7.11 Run Mode Clock Configuration (RCC) Register

Figure 1-103. Run Mode Clock Configuration (RCC) Register

31	28	27	26	0
Reserved		ACG	Reserved	
R-0:0		R/W-0	R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-114. Run Mode Clock Configuration (RCC) Register Field Descriptions

Bit	Field	Value	Description
31-28	Reserved		Reserved
27	ACG		Auto Clock Gating This bit specifies whether the system uses the Sleep-Mode Clock Gating Control (SCGCx) registers and Deep-Sleep-Mode Clock Gating Control (DCGC) registers if the M3 CPU enters a sleep or deep-sleep mode. The RCGC registers are always used to control the clocks in run mode.
		0	The Run-Mode Clock Gating Control (RCGCx) registers are used when the microcontroller enters a sleep mode.
		1	The SCGCx or DCGCx registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The SCGCx and DCGCx registers allow unused peripherals to consume less power when the M3 CPU in a sleep mode.
26-0	Reserved		Reserved

1.13.7.12 Master GPIO High Performance Bus Control (GPIOHBCTL) Register

Figure 1-104. Master GPIO High Performance Bus Control (GPIOHBCTL) Register

31	16
Reserved	
R-0	

15	9	8	7	6	5	4	3	2	1	0
Reserved		PORTJ	PORTH	PORTG	PORTF	PORTE	PORTD	PORTC	PORTB	PORTA
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-115. Master GPIO High Performance Bus Control (GPIOHBCTL) Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	PORT J		PORT J AHB. This bit defines the memory aperture for Port J
		0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
		1	Advanced High-Performance Bus (AHB)
7	PORT H		PORT H AHB. This bit defines the memory aperture for Port H
		0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
		1	Advanced High-Performance Bus (AHB)
6	PORT G		PORT G AHB. This bit defines the memory aperture for Port G
		0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
		1	Advanced High-Performance Bus (AHB)
5	PORT F		PORT F AHB. This bit defines the memory aperture for Port F
		0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
		1	Advanced High-Performance Bus (AHB)
4	PORT E		PORT E AHB. This bit defines the memory aperture for Port E
		0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
		1	Advanced High-Performance Bus (AHB)
3	PORT D		PORT D AHB. This bit defines the memory aperture for Port D
		0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
		1	Advanced High-Performance Bus (AHB)
2	PORT C		PORT C AHB. This bit defines the memory aperture for Port C
		0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
		1	Advanced High-Performance Bus (AHB)
1	PORT B		PORT B AHB. This bit defines the memory aperture for Port B
		0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
		1	Advanced High-Performance Bus (AHB)
0	PORT A		PORT A AHB. This bit defines the memory aperture for Port A
		0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
		1	Advanced High-Performance Bus (AHB)

1.13.7.13 Run Mode Clock Gating Control Register 0 (RCGC0)

Figure 1-105. Run Mode Clock Gating Control Register 0 (RCGC0)

31 29	28	27 4	3	2 0
Reserved	WDT1	Reserved	WDT0	Reserved
R-0:0	R/W-0	R-0:0	R/W-0	R-0:0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-116. Run Mode Clock Gating Control Register 0 (RCGC0) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
28	WDT1		WDT1 Clock Gating Control This bit controls the clock gating for the WDT1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
27-4	Reserved		Reserved
3	WDT0		WDT0 Clock Gating Control This bit controls the clock gating for the WDT0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-116. Run Mode Clock Gating Control Register 0 (RCGC0) Field Descriptions (continued)

Bit	Field	Value	Description
2-0	Reserved		Reserved

1.13.7.14 Sleep Mode Clock Gating Control Register 0 (SCGC0)

Figure 1-106. Sleep Mode Clock Gating Control Register 0 (SCGC0)

31 29	28	27 4	3	2 0
Reserved	WDT1	Reserved	WDT0	Reserved
R-0:0	R/W-0	R-0:0	R/W-0	R-0:0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-117. Sleep Mode Clock Gating Control Register 0 (SCGC0) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
28	WDT1		WDT1 Clock Gating Control in Sleep Mode This bit controls the clock gating for the WDT1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
27-4	Reserved		Reserved
3	WDT0		WDT0 Clock Gating Control in Sleep Mode This bit controls the clock gating for the WDT0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
2-0	Reserved		Reserved

1.13.7.15 Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

Figure 1-107. Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

31 29	28	27 4	3	2 0
Reserved	WDT1	Reserved	WDT0	Reserved
R-0:0	R/W-0	R-0:0	R/W-0	R-0:0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-118. Deep Sleep Mode Clock Gating Control Register 0 (DCGC0) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
28	WDT1		WDT1 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the WDT1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
27-4	Reserved		Reserved
3	WDT0		WDT0 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the WDT0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
2-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.7.16 Run Mode Clock Gating Control Register 1 (RCGC1)

Figure 1-108. Run Mode Clock Gating Control Register 1 (RCGC1)

31	30	29	28	27	26	25	24
Reserved	EPI	Reserved					
R-0	R/W-0	R-0					

23	22	21	20	19	18	17	16
Reserved				TIMER3	TIMER2	TIMER1	TIMER0
R-0				R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
Reserved	I2C1	Reserved	I2C0	Reserved			
R-0	R/W-0	R-0	R/W-0	R-0			

7	6	5	4	3	2	1	0
SSI3	SSI2	SSI1	SSI0	UART3	UART2	UART1	UART0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-119. Run Mode Clock Gating Control Register 1 (RCGC1) Field Descriptions

Bit	Field	Value	Description
31	Reserved		Reserved
30	EPI		EPI Clock Gating Control in Run Mode This bit controls the clock gating for the EPI module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
29-20	Reserved		Reserved
19	TIMER3		GPT3 Clock Gating Control in Run Mode This bit controls the clock gating for the TIMER3 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
18	TIMER2		GPT2 Clock Gating Control in Run Mode This bit controls the clock gating for the TIMER2 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
17	TIMER1		GPT1 Clock Gating Control in Run Mode This bit controls the clock gating for the TIMER1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
16	TIMER0		GPT0 Clock Gating Control in Run Mode This bit controls the clock gating for the TIMER0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
15	Reserved		Reserved
14	I2C1		I2C1 Clock Gating Control in Run Mode This bit controls the clock gating for the I2C1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
13	Reserved		Reserved
12	I2C0		I2C0 Clock Gating Control in Run Mode This bit controls the clock gating for the I2C0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
11-8	Reserved		Reserved
7	SSI3		SSI3 Clock Gating Control in Run Mode This bit controls the clock gating for the SSI3 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-119. Run Mode Clock Gating Control Register 1 (RCGC1) Field Descriptions (continued)

Bit	Field	Value	Description
6	SSI2		SSI2 Clock Gating Control in Run Mode This bit controls the clock gating for the SSI2 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
5	SSI1		SSI1 Clock Gating Control in Run Mode This bit controls the clock gating for the SSI1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
4	SSI0		SSI0 Clock Gating Control in Run Mode This bit controls the clock gating for the SSI0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
3	UART3		UART3 Clock Gating Control in Run Mode This bit controls the clock gating for the UART3 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
2	UART2		UART2 Clock Gating Control in Run Mode This bit controls the clock gating for the UART2 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
1	UART1		UART1 Clock Gating Control in Run Mode This bit controls the clock gating for the UART1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
0	UART0		UART0 Clock Gating Control in Run Mode This bit controls the clock gating for the UART0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

1.13.7.17 Sleep Mode Clock Gating Control Register 1 (SCGC1)

Figure 1-109. Sleep Mode Clock Gating Control Register 1 (SCGC1)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved				TIMER3	TIMER2	TIMER1	TIMER0
R-0				R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
Reserved	I2C1	Reserved	I2C0	Reserved			
R-0	R/W-0	R-0	R/W-0	R-0			

7	6	5	4	3	2	1	0
SSI3	SSI2	SSI1	SSI0	UART3	UART2	UART1	UART0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-120. Sleep Mode Clock Gating Control Register 1 (SCGC1) Field Descriptions

Bit	Field	Value	Description
31	Reserved		Reserved
30	EPI		EPI Clock Gating Control in Sleep Mode This bit controls the clock gating for the EPI module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
29-20	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-120. Sleep Mode Clock Gating Control Register 1 (SCGC1) Field Descriptions (continued)

Bit	Field	Value	Description
19	TIMER3		GPT3 Clock Gating Control in Sleep Mode This bit controls the clock gating for the TIMER3 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
18	TIMER2		GPT2 Clock Gating Control in Sleep Mode This bit controls the clock gating for the TIMER2 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
17	TIMER1		GPT1 Clock Gating Control in Sleep Mode This bit controls the clock gating for the TIMER1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
16	TIMER0		GPT0 Clock Gating Control in Sleep Mode This bit controls the clock gating for the TIMER0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
15	Reserved		Reserved
14	I2C1		I2C1 Clock Gating Control in Sleep Mode This bit controls the clock gating for the I2C1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
13	Reserved		Reserved
12	I2C0		I2C0 Clock Gating Control in Sleep Mode This bit controls the clock gating for the I2C0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
11-8	Reserved		Reserved
7	SSI3		SSI3 Clock Gating Control in Sleep Mode This bit controls the clock gating for the SSI3 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
6	SSI2		SSI2 Clock Gating Control in Sleep Mode This bit controls the clock gating for the SSI2 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
5	SSI1		SSI1 Clock Gating Control in Sleep Mode This bit controls the clock gating for the SSI1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
4	SSI0		SSI0 Clock Gating Control in Sleep Mode This bit controls the clock gating for the SSI0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
3	UART3		UART3 Clock Gating Control in Sleep Mode This bit controls the clock gating for the UART3 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
2	UART2		UART2 Clock Gating Control in Sleep Mode This bit controls the clock gating for the UART2 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
1	UART1		UART1 Clock Gating Control in Sleep Mode This bit controls the clock gating for the UART1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
0	UART0		UART0 Clock Gating Control in Sleep Mode This bit controls the clock gating for the UART0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

Copyright © 2012–2019, Texas Instruments Incorporated

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

Figure 1-110. Deep Sleep Mode Clock Gating Control Register 1 (DCGC1)

31	30	29	24
	EPI		

23	20	19	18	17	16
Reserved		TIMER3	TIMER2	TIMER1	TIMER0
R-0		R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	11	8
Reserved	I2C1	Reserved	I2C0	Reserved	
R-0	R/W-0	R-0	R/W-0	R-0	

7	6	5	4	3	2	1	0
SSI3	SSI2	SSI1	SSI0	UART3	UART2	UART1	UART0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-121. Deep Sleep Mode Clock Gating Control Register 1 (DCGC1) Field Descriptions

Bit	Field	Value	Description
31	Reserved		Reserved
30	EPI		EPI Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the EPI module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
29-20	Reserved		Reserved
19	TIMER3		GPT3 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the TIMER3 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
18	TIMER2		GPT2 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the TIMER2 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
17	TIMER1		GPT1 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the TIMER1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
16	TIMER0		GPT0 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the TIMER0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
15	Reserved		Reserved
14	I2C1		I2C1 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the I2C1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
13	Reserved		Reserved
12	I2C0		I2C0 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the I2C0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
11-8	Reserved		Reserved
7	SSI3		SSI3 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the SSI3 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-121. Deep Sleep Mode Clock Gating Control Register 1 (DCGC1) Field Descriptions (continued)

Bit	Field	Value	Description
6	SSI2		SSI2 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the SSI2 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
5	SSI1		SSI1 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the SSI1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
4	SSI0		SSI0 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the SSI0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
3	UART3		UART3 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the UART3 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
2	UART2		UART2 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the UART2 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
1	UART1		UART1 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the UART1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
0	UART0		UART0 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the UART0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

1.13.7.19 Run Mode Clock Gating Control Register 2 (RCGC2)

Figure 1-111. Run Mode Clock Gating Control Register 2 (RCGC2)

31	30	29	28	27	26	25	24
Reserved			EMAC0	Reserved			
R-0			R/W-0	R-0			
23	22	21	20	19	18	17	16
Reserved							USB
R-0							R/W-0
15	14	13	12	11	10	9	8
Reserved		µDMA	Reserved				GPIOJ
R-0		R/W-0	R-0				R/W-0
7	6	5	4	3	2	1	0
GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-122. Run Mode Clock Gating Control Register 2 (RCGC2) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
28	EMAC0		EMAC0 Clock Gating Control in Run Mode This bit controls the clock gating for the EMAC0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
27-17	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-122. Run Mode Clock Gating Control Register 2 (RCGC2) Field Descriptions (continued)

Bit	Field	Value	Description
16	USB		USB0 Clock Gating Control in Run Mode This bit controls the clock gating for the UDB module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
15-14	Reserved		Reserved
13	µDMA		µDMA Clock Gating Control in Run Mode This bit controls the clock gating for the µDMA module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
12-9	Reserved		Reserved
8	GPIOJ		GPIOJ Clock Gating Control in Run Mode This bit controls the clock gating for the GPIOJ module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
7	GPIOH		GPIOH Clock Gating Control in Run Mode This bit controls the clock gating for the GPIOH module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
6	GPIOG		GPIOG Clock Gating Control in Run Mode This bit controls the clock gating for the GPIOG module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
5	GPIOF		GPIOF Clock Gating Control in Run Mode This bit controls the clock gating for the GPIOF module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
4	GPIOE		GPIOE Clock Gating Control in Run Mode This bit controls the clock gating for the GPIOE module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
3	GPIOD		GPIOD Clock Gating Control in Run Mode This bit controls the clock gating for the GPIOD module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
2	GPIOC		GPIOC Clock Gating Control in Run Mode This bit controls the clock gating for the GPIOC module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
1	GPIOB		GPIOB Clock Gating Control in Run Mode This bit controls the clock gating for the GPIOB module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
0	GPIOA		GPIOA Clock Gating Control in Run Mode This bit controls the clock gating for the GPIOA module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.7.20 Sleep Mode Clock Gating Control Register 2 (SCGC2)

Figure 1-112. Sleep Mode Clock Gating Control Register 2 (SCGC2)

31	30	29	28	27	26	25	24
Reserved			EMAC0	Reserved			
R-0			R/W-0	R-0			

23	22	21	20	19	18	17	16
Reserved							USB
R-0							R/W-0

15	14	13	12	11	10	9	8
Reserved		µDMA	Reserved				GPIOJ
R-0		R/W-0	R-0				R/W-0

7	6	5	4	3	2	1	0
GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-123. Sleep Mode Clock Gating Control Register 2 (SCGC2) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
28	EMAC0		EMAC0 Clock Gating Control in Sleep Mode This bit controls the clock gating for the EMAC0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
27-17	Reserved		Reserved
16	USB		USB0 Clock Gating Control in Sleep Mode This bit controls the clock gating for the USB module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
15-14	Reserved		Reserved
13	µDMA		µDMA Clock Gating Control in Sleep Mode This bit controls the clock gating for the µDMA module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
12-9	Reserved		Reserved
8	GPIOJ		GPIOJ Clock Gating Control in Sleep Mode This bit controls the clock gating for the GPIOJ module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
7	GPIOH		GPIOH Clock Gating Control in Sleep Mode This bit controls the clock gating for the GPIOH module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
6	GPIOG		GPIOG Clock Gating Control in Sleep Mode This bit controls the clock gating for the GPIOG module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
5	GPIOF		GPIOF Clock Gating Control in Sleep Mode This bit controls the clock gating for the GPIOF module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
4	GPIOE		GPIOE Clock Gating Control in Sleep Mode This bit controls the clock gating for the GPIOE module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-123. Sleep Mode Clock Gating Control Register 2 (SCGC2) Field Descriptions (continued)

Bit	Field	Value	Description
3	GPIOD		GPIOD Clock Gating Control in Sleep Mode This bit controls the clock gating for the GPIOD module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
2	GPIOC		GPIOC Clock Gating Control in Sleep Mode This bit controls the clock gating for the GPIOC module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
1	GPIOB		GPIOB Clock Gating Control in Sleep Mode This bit controls the clock gating for the GPIOB module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
0	GPIOA		GPIOA Clock Gating Control in Sleep Mode This bit controls the clock gating for the GPIOAmodule. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.7.21 Deep Sleep Mode Clock Gating Control Register 2 (DCGC2)

Figure 1-113. Deep Sleep Mode Clock Gating Control Register 2 (DCGC2)

31	29	28	27	24
Reserved		EMAC0	Reserved	
R-0		R/W-0	R-0	

23	17	16
Reserved		
R-0		

15	14	13	12	9	8
Reserved		µDMA	Reserved		GPIOJ
R-0		R/W-0	R-0		

7	6	5	4	3	2	1	0
GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-124. Deep Sleep Mode Clock Gating Control Register 2 (DCGC2) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
28	EMAC0		EMAC0 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the EMAC0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
27-17	Reserved		Reserved
16	USB		USB0 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the USB module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
15-14	Reserved		Reserved
13	µDMA		µDMA Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the µDMA module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
12-9	Reserved		Reserved
8	GPIOJ		GPIOJ Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the GPIOJ module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
7	GPIOH		GPIOH Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the GPIOH module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
6	GPIOG		GPIOG Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the GPIOG module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
5	GPIOF		GPIOF Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the GPIOF module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
4	GPIOE		GPIOE Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the GPIOE module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
3	GPIOD		GPIOD Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the GPIOD module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-124. Deep Sleep Mode Clock Gating Control Register 2 (DCGC2) Field Descriptions (continued)

Bit	Field	Value	Description
2	GPIOC		GPIOC Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the GPIOC module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
1	GPIOB		GPIOB Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the GPIOB module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
0	GPIOA		GPIOA Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the GPIOA module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

1.13.7.22 Run Mode Clock Gating Control Register 3 (RCGC3)

Figure 1-114. Run Mode Clock Gating Control Register 3 (RCGC3)

31	26	25	24	23	16
Reserved		CAN1	CAN0	Reserved	
R-0:0		R/W-0	R/W-0	R-0:0	

15	1	0
Reserved		UART4
R-0:0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-125. Run Mode Clock Gating Control Register 3 (RCGC3) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25	CAN1		CAN1 Clock Gating Control in Run Mode This bit controls the clock gating for the CAN1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
24	CAN0		CAN0 Clock Gating Control in Run Mode This bit controls the clock gating for the CAN0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
23-1	Reserved		Reserved
0	UART4		UART4 Clock Gating Control in Run Mode This bit controls the clock gating for the UART4 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

1.13.7.23 Sleep Mode Clock Gating Control Register 3 (SCGC3)

Figure 1-115. Sleep Mode Clock Gating Control Register 3 (SCGC3)

31	26	25	24	23	16
Reserved		CAN1	CAN0	Reserved	
R-0:0		R/W-0	R/W-0	R-0:0	

15	1	0
Reserved		UART4
R-0:0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-126. Sleep Mode Clock Gating Control Register 3 (SCGC3) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25	CAN1		CAN1 Clock Gating Control in Sleep Mode This bit controls the clock gating for the CAN1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
24	CAN0		CAN0 Clock Gating Control in Sleep Mode This bit controls the clock gating for the CAN0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
23-1	Reserved		Reserved
0	UART4		UART4 Clock Gating Control in Sleep Mode This bit controls the clock gating for the UART4 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

1.13.7.24 Deep Sleep Mode Clock Gating Control Register 3 (DCGC3)

Figure 1-116. Deep Sleep Mode Clock Gating Control Register 3 (DCGC3)

31	26	25	24	23	16
Reserved		CAN1	CAN0	Reserved	
R-0:0		R/W-0	R/W-0	R-0:0	

15	1	0
Reserved		UART4
R-0:0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-127. Deep Sleep Mode Clock Gating Control Register 3 (DCGC3) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25	CAN1		CAN1 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the CAN1 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
24	CAN0		CAN0 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the CAN0 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.
23-1	Reserved		Reserved
0	UART4		UART4 Clock Gating Control in Deep Sleep Mode This bit controls the clock gating for the UART4 module. If set, the module receives a clock and functions. Otherwise, it is unclocked and disabled. If the module is unclocked, reads or writes to the module generates bus faults.

1.13.7.25 Deep Sleep Clock Configuration (DSLPCLKCFG) Register

NOTE: M3 Watchdog 1 will be clocked by the deep sleep clock selected by the DSLPCLKCFG.DSOSCSRC bits.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 1-117. Deep Sleep Clock Configuration (DSLPCLKCFG) Register

31	27	26	23	22	16
Reserved		DSDIVOVRIDE		Reserved	
R-0:0		R/W-0		R-0	

15	7	6	4	3	0
Reserved		DSOSCSRC		Reserved	
R-0		R/W-0:0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-128. Deep Sleep Clock Configuration (DSLPCLKCFG) Register Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26-23	DSDIVOVRIDE		Deep Sleep Divider Override Divider field override in deep sleep mode. 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 M3SSDIVSEL register. This divider is applied to the source selected by the DSOSCSRC field.
		0x0	/1
		0x1	/2
		0x2	/3
		0x3	/4
		0x4	/5
		0x5	/6
		0x6	/7
		0x7	/8
		0x8	/9
		0x9	/10
		0xA	/11
		0xB	/12
		0xC	/13
		0xD	/14
		0xE	/15
		0xF	/16
22-7	Reserved		Reserved
6-4	DSOSCSRC		Deep Sleep Mode Clock Source Select
		0x0	Use OSCCLK as the clock source
		0x1	Use the 32-kHz clock as the clock source
		0x2	Use the 10-MHz ATOB clock as the clock source
		0x3-0x7	Reserved
3-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.7.26 C28 CPU Timer 2 Clock Configuration (CLKCTL) Register

Figure 1-118. C28 CPU Timer 2 Clock Configuration (CLKCTL) Register

15	8
Reserved	
R-0:0	

7	5	4	3	2	0
TMR2CLKPRESCALE		TMR2CLKSRCSEL		Reserved	
R/W-0		R/W-0		R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-129. C28 CPU Timer 2 Clock Configuration (CLKCTL) Register Field Descriptions

Bit	Field	Value	Description
15-8	Reserved		Reserved
7-5	TMR2CLKPRESCALE		Timer2 Clock Prescale Select These bits select the pre-scale value for the selected clock source for CPU Timer 2. This selection is not affected by the missing clock detect circuit.
		0,0,0	/1 (default on reset)
		0,0,1	/2
		0,1,0	/4
		0,1,1	/8
		1,0,0	/16
		1,0,1	Reserved
		1,1,0	Reserved
		1,1,1	Reserved
4-3	TMR2CLKSRCSEL		Timer2 Clock Source Select This bit selects the source for C28 CPU Timer 2. This selection is not affected by the missing clock detect circuit.
		00	C28 SYSCLK selected (default on reset, pre-scaler is bypassed)
		01	External oscillator (X1) selected
		10	OSCCLK selected
		11	Reserved
2-0	Reserved		Reserved

1.13.7.27 Peripheral Clock Control Register 0 (PCLKCR0)

Figure 1-119. Peripheral Clock Control Register 0 (PCLKCR0)

15	13	12	11	10	9	8
Reserved		MCBSPAENCLK	Reserved	SCIAENCLK	Reserved	SPIAENCLK
R-0:0		R/W-0	R-0	R/W-0	R-0:0	R/W-0

7	5	4	3	2	1	0
Reserved		I2CAENCLK	Reserved	TBCLKSYNC	Reserved	HRPWMENCLK
R-0:0		R/W-0	R-0:0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-130. Peripheral Clock Control Register 0 (PCLKCR0) Register Field Descriptions

Bit	Field	Value	Description
15-3	Reserved		Reserved
12	MENCLK		McBSP-A Clock Enable When set, this enables the clock to the McBSP-A module.
		0	McBSP-A clock is disabled
		1	McBSP-A clock is enabled
11	Reserved		Reserved
10	SCIAENCLK		SCI-A Clock Enable When set, this enables the clock to the C28 SCI-A module.
		0	SCI-A clock is disabled
		1	SCI-A clock is enabled
9	Reserved		Reserved
8	SPIAENCLK		SPI-A Clock EnableWhen set, this enables the clock to the C28 SPI-A module.
		0	SPI-A clock is disabled
		1	SPI-A clock is enabled
7-5	Reserved		Reserved
4	I2CENCLK		I2C-A Clock Enable When set, this enables the clock to the C28 I2C-A module.
		0	I2C-A clock is disabled
		1	I2C-A clock is enabled
3	Reserved		Reserved
2	TBCLKSYNC		ePWM Clock Sync When set PWM time bases of all modules start counting.
1	Reserved		Reserved
0	HRPWMENCLK		HRPWM Clock Enable When set, this enables the clock to the HRPWM module.
		0	HRPWM clock is disabled
		1	HRPWM clock is enabled

1.13.7.28 Peripheral Clock Control Register 1 (PCLKCR1)

Figure 1-120. Peripheral Clock Control Register 1 (PCLKCR1)

15	14	13	12	11	10	9	8
EQEP2ENCLK	EQEP1ENCLK	ECAP6ENCLK	ECAP5ENCLK	ECAP4ENCLK	ECAP3ENCLK	ECAP2ENCLK	ECAP1ENCLK
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
EPWM8ENCLK	EPWM7ENCLK	EPWM6ENCLK	EPWM5ENCLK	EPWM4ENCLK	EPWM3ENCLK	EPWM2ENCLK	EPWM1ENCLK
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-131. Peripheral Clock Control Register 1 (PCLKCR1) Register Field Descriptions

Bit	Field	Value	Description
15-14	EQEPxENCLK (n = 2-1)		eQEP2-1 Clock Enables When set, this enables the clock to the respective eQEP module.
		0	Clock is disabled
		1	Clock is enabled
13-8	ECAPxENCLK (n = 6-1)		eCAP6-1 Clock Enables When set, this enables the clock to the respective eCAP module.
		0	Clock is disabled
		1	Clock is enabled

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-131. Peripheral Clock Control Register 1 (PCLKCR1) Register Field Descriptions (continued)

Bit	Field	Value	Description
7-0	EPWMxENCLK (n = 8-1)		ePWM8-1 Clock Enables When set, this enables the clock to the respective ePWM module.
		0	Clock is disabled
		1	Clock is enabled

1.13.7.29 Peripheral Clock Control Register 2 (PCLKCR2)

Figure 1-121. Peripheral Clock Control Register 2 (PCLKCR2)

15	9	8
Reserved		EQEP3ENCLK
R-0:0		R/W-0

7	1	0
Reserved		EPWM9ENCLK
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-132. Peripheral Clock Control Register 2 (PCLKCR2) Register Field Descriptions

Bit	Field	Value	Description
15-9	Reserved		Reserved
8	EQEP3ENCLK		eQEP3 Clock Enable When set, this enables the clock to the eQEP3 module.
		0	eQEP3 clock is disabled
		1	eQEP3 clock is enabled
7-1	Reserved		Reserved
0	EPWM9ENCLK		ePWM9 Clock Enable When set, this enables the clock to the ePWM9 module.
		0	ePWM9 clock is disabled
		1	ePWM9 clock is enabled

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.7.30 Peripheral Clock Control Register 3 (PCLKCR3)

Figure 1-122. Peripheral Clock Control Register 3 (PCLKCR3)

15	12	11	10	9	8
Reserved		DMAENCLK	CPUTIMER2ENCLK	CPUTIMER1ENCLK	CPUTIMER0ENCLK
R-0		R/W-0	R/W-1	R/W-1	R/W-1

7	0
Reserved	
R-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-133. Peripheral Clock Control Register 3 (PCLKCR3) Register Field Descriptions

Bit	Field	Value	Description
15-12	Reserved		Reserved
11	DMAENCLK		C28 DMA Clock Enable When set, this enables the clock to the CPU timers on the C28 subsystem.
		0	C28 DMA clock is disabled
		1	C28 DMA clock is enabled
10-8	CPUTIMERnENCLK (n = 2-0)		C28 CPU Timer 2-0 Clock Enables When set, this enables the clock to the CPU timers on the C28 subsystem.
		0	Timer clock is disabled
		1	Timer clock is enabled
7-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.7.31 High-Speed Clock Prescaler (CHISPCP) Register

Figure 1-123. High-Speed Clock Prescaler (CHISPCP) Register

15	3	2	0
Reserved		HSPCLK	
R-0		R/W-001	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-134. High-Speed Clock Prescaler (CHISPCP) Register Field Descriptions

Bit	Field	Value	Description
15-3	Reserved		Reserved
2-0	HSPCLK		High-Speed Clock Prescaler These bits configure the high-speed peripheral clock (HSPCLK) rate relative to the C28 SYSCLKOUT:
		000	HSPCLK = SYSCLKOUT / 1
		001	HSPCLK = SYSCLKOUT / 2
		010	HSPCLK = SYSCLKOUT / 4
		011	HSPCLK = SYSCLKOUT / 6
		100	HSPCLK = SYSCLKOUT / 8
		101	HSPCLK = SYSCLKOUT / 10
		110	HSPCLK = SYSCLKOUT / 12

1.13.7.32 Low-Speed Clock Prescaler (CLOSPCP) Register

Figure 1-124. Low-Speed Clock Prescaler (CLOSPCP) Register

15	3	2	0
Reserved		LSPCLK	
R-0		R/W-010	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-135. Low-Speed Clock Prescaler (CLOSPCP) Register Field Descriptions

Bit	Field	Value	Description
15-3	Reserved		Reserved
2-0	LSPCLK		Low-Speed Clock Prescaler These bits configure the low-speed peripheral clock (LSPCLK) rate relative to the C28 SYSCLKOUT:
		000	LSPCLK = SYSCLKOUT / 1
		001	LSPCLK = SYSCLKOUT / 2
		010	LSPCLK = SYSCLKOUT / 4
		011	LSPCLK = SYSCLKOUT / 6
		100	LSPCLK = SYSCLKOUT / 8
		101	LSPCLK = SYSCLKOUT / 10
		110	LSPCLK = SYSCLKOUT / 12
		111	LSPCLK = SYSCLKOUT / 14

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.7.33 C28 XCLKOUT Divider Register (CXCLK)

Figure 1-125. C28 XCLKOUT Divider Register (CXCLK)

15	2	1	0
Reserved		XCLKOUTDIV	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-136. C28 XCLKOUT Divider Register (CXCLK) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved		Reserved
1-0	XCLKOUTDIV		XCLKOUT Divider Register. These two bits select the XCLKOUT frequency ratio relative to the C28 SYSCLKOUT. The ratios are given below.
		000	XCLKOUT = C28 SYSCLKOUT/4
		001	XCLKOUT = C28 SYSCLKOUT/2
		010	XCLKOUT = C28 SYSCLKOUT
		011	XCLKOUT = Off

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.8 Master Subsystem Code Security Module (CSM) Registers

1.13.8.1 Z1_CSMKEY0 Register

Figure 1-126. Z1_CSMKEY0 Register

31	0
CSMKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-137. Z1_CSMKEY0 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unlock M3 zone1, write the same value in CSMPSWD0 of zone1 to this register.

1.13.8.2 Z1_CSMKEY1 Register

Figure 1-127. Z1_CSMKEY1 Register

31	0
CSMKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-138. Z1_CSMKEY1 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unlock M3 zone1, write the same value in CSMPSWD1 of zone1 to this register.

1.13.8.3 Z1_CSMKEY2 Register

Figure 1-128. Z1_CSMKEY2 Register

31	0
CSMKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-139. Z1_CSMKEY2 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unlock M3 zone1, write the same value in CSMPSWD2 of zone1 to this register.

1.13.8.4 Z1_CSMKEY3 Register

Figure 1-129. Z1_CSMKEY3 Register

31	0
CSMKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-140. Z1_CSMKEY3 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unclock M3 zone1, write the same value in CSMPSWD3 of zone1 to this register.

1.13.8.5 Z1_ECSLKEY0 Register

Figure 1-130. Z1_ECSLKEY0 Register

31	0
ECSLKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-141. Z1_ECSLKEY0 Register Field Descriptions

Bit	Field	Value	Description
31-0	ECSLKEY		To disable ECSL logic active on the M3 zone1, write the same value in ECSLPSWD0 of zone1 to this register.

1.13.8.6 Z1_ECSLKEY1 Register

Figure 1-131. Z1_ECSLKEY1 Register

31	0
ECSLKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-142. Z1_ECSLKEY1 Register Field Descriptions

Bit	Field	Value	Description
31-0	ECSLKEY		To disable ECSL logic active on the M3 zone1, write the same value in ECSLPSWD1 of zone1 to this register.

1.13.8.7 Z2_CSMKEY0 Register

Figure 1-132. Z2_CSMKEY0 Register

31	0
CSMKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

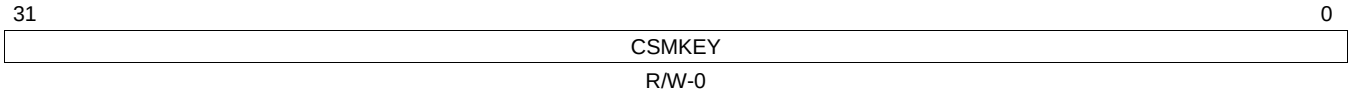
Table 1-143. Z2_CSMKEY0 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unlock M3 zone2, write the same value in CSMPSWD0 of zone2 to this register.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.8.8 Z2_CSMKEY1 Register

Figure 1-133. Z2_CSMKEY1 Register



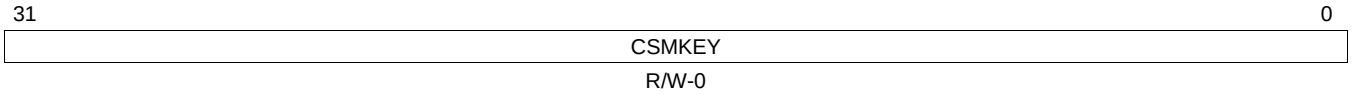
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-144. Z2_CSMKEY1 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unlock M3 zone2, write the same value in CSMPSWD1 of zone2 to this register.

1.13.8.9 Z2_CSMKEY2 Register

Figure 1-134. Z2_CSMKEY2 Register



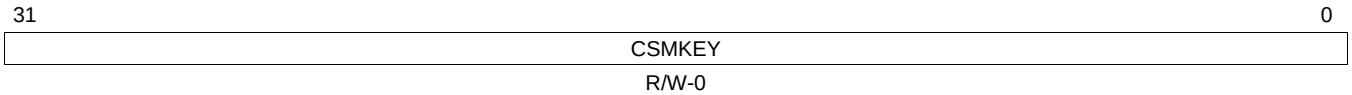
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-145. Z2_CSMKEY2 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unlock M3 zone2, write the same value in CSMPSWD2 of zone2 to this register.

1.13.8.10 Z2_CSMKEY3 Register

Figure 1-135. Z2_CSMKEY3 Register



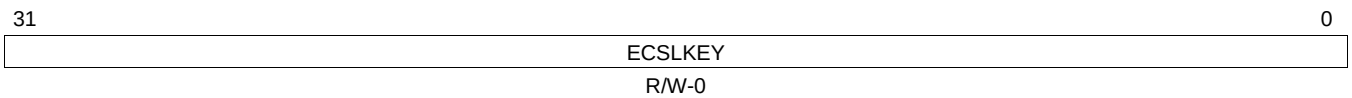
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-146. Z2_CSMKEY3 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unlock M3 zone2, write the same value in CSMPSWD3 of zone2 to this register.

1.13.8.11 Z2_ECSLKEY0 Register

Figure 1-136. Z2_ECSLKEY0 Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-147. Z2_ECSLKEY0 Register Field Descriptions

Bit	Field	Value	Description
31-0	ECSLKEY		To disable ECSL logic active on the M3 zone2, write the same value in ECSLPSWD0 of zone2 to this register.

1.13.8.12 Z2_ECSLKEY1 Register

Figure 1-137. Z2_ECSLKEY1 Register

31	0
ECSLKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-148. Z2_ECSLKEY1 Register Field Descriptions

Bit	Field	Value	Description
31-0	ECSLKEY		To disable ECSL logic active on the M3 zone2, write the same value in ECSLPSWD1 of zone2 to this register.

1.13.8.13 Z1_CSMCR Register

Figure 1-138. Z1_CSMCR Register

31	24
Reserved	
R-0	

23	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
FORCESEC	Reserved		ECSL-ARMED	CSM-ARMED	ECSL-MATCH	CSM-MATCH	ECSL-ALLONE
R=0/W=1	R-0		R-0	R-0	R-x	R-x	R-x

7	6	5	4	0
CSM-ALLONE	ECSL-ALLZERO	CSM-ALLZERO	Reserved	
R-x	R-x	R-x	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-149. Z1_CSMCR Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	FORCESEC	0	Reads always return 0 (Self clear bit). Clears the KEY registers and makes M3 zone1 secure. The password match flow described in Section 1.10.3.2 must be followed to unsecure zone1 again.
		1	Writing a 1 to this bit clears all the KEY (CSM & ECSL both) registers for M3 zone1 and makes zone1 secure if CSMPSWD are not all 1's.
14-13	Reserved		Reserved
12	ECSL-ARMED	0	Dummy read to M3-Zone1 ECSL PWL (Password locations) have not been performed.
		1	Dummy read to M3-Zone1 ECSL PWL (Password locations) have been performed.
11	CSM-ARMED	0	Dummy read to M3-Zone1 CSM PWL (Password locations) have not been performed.
		1	Dummy read to M3-Zone1 CSM PWL (Password locations) have been performed.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-149. Z1_CSMCR Register Field Descriptions (continued)

Bit	Field	Value	Description
10	ECSL-MATCH		Status bit indicate if ECSL is enabled or disabled for M3 zone1. This is valid only when ECSL_ARMED = 1.
		0	M3 Zone1 ECSL is disabled
		1	M3 Zone1 ECSL is enabled
9	CSM-MATCH		Status bit that reflects the security state of Zone1. This is valid only when CSM_ARMED = 1.
		0	M3 Zone1 is secure if CSM-ALLONE is 0
		1	M3 Zone1 is non-secure if CSM-ALLZERO is 0.
8	ECSL-ALLONE		Shows the state of ECSL Password programmed in Flash. This is valid only when ECSL_ARMED = 1.
		0	ECSL Password of M3 Zone1 does not contain all 1's
		1	ECSL Password of M3 Zone1 contains all 1's
7	CSM-ALLONE		Shows the state of CSM Password programmed in Flash. This is valid only when CSM_ARMED = 1.
		0	CSM Passwords of M3 Zone1 does not contain all 1's
		1	CSM Passwords of M3 Zone1 contains all 1's
6	ECSL-ALLZERO		Shows the state of ECSL Password programmed in Flash. This is valid only when ECSL_ARMED = 1.
		0	ECSL Password of M3 Zone1 does not contain all 0's
		1	ECSL Password of M3 Zone1 contains all 0's
5	CSM-ALLZERO		Shows the state of CSM Password programmed in Flash. This is valid only when CSM_ARMED = 1.
		0	CSM Passwords of M3 Zone1 does not contain all 0's
		1	CSM Passwords of M3 Zone1 contains all 0's
4-0	Reserved		Reserved

1.13.8.14 Z2_CSMCR Register

Figure 1-139. Z2_CSMCR Register

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
R-0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FORCESEC	Reserved		ECSL-ARMED	CSM-ARMED	ECSL-MATCH	CSM-MATCH	ECSL-ALLONE
R=0/W=1	R-0		R-0	R-0	R-x	R-x	R-x
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CSM-ALLONE	ECSL-ALLZERO	CSM-ALLZERO	Reserved				
R-x	R-x	R-x	R-0				

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-150. Z2_CSMCR Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	FORCESEC	0	Reads always return 0 (Self clear bit). Clears the KEY registers and makes M3 zone2 secure. The password match flow described in Section 1.10.3.2 must be followed to unsecure the zone2 again.
		1	Writing a 1 to this bit clears all the KEY (CSM & ECSL both) registers for M3 zone2 and makes zone2 secure if CSMPSWD are not all 1's.
14-13	Reserved		Reserved
12	ECSL-ARMED	0	Dummy read to M3-Zone2 ECSL PWL (Password locations) have not been performed.
		1	Dummy read to M3-Zone2 ECSL PWL (Password locations) have been performed.
11	CSM-ARMED	0	Dummy read to M3-Zone2 CSM PWL (Password locations) have not been performed.
		1	Dummy read to M3-Zone2 CSM PWL (Password locations) have been performed.
10	ECSL-MATCH		Status bit indicate if ECSL is enabled or disabled for M3 zone2. . This is valid only when ECSL_ARMED = 1.
		0	M3 Zone2 ECSL is disabled
		1	M3 Zone2 ECSL is enabled
9	CSM-MATCH		Status bit that reflects the security state of Zone2. This is valid only when CSM_ARMED = 1.
		0	M3 Zone2 is secure if CSM-ALLONE is 0
		1	M3 Zone2 is non-secure if CSM-ALLZERO is 0.
8	ECSL-ALLONE		Shows the state of ECSL Password programmed in Flash. This is valid only when ECSL_ARMED = 1.
		0	ECSL Password of M3 Zone2 does not contain all 1's
		1	ECSL Password of M3 Zone2 contains all 1's
7	CSM-ALLONE		Shows the state of CSM Password programmed in Flash. This is valid only when CSM_ARMED = 1.
		0	CSM Passwords of M3 Zone2 does not contain all 1's
		1	CSM Passwords of M3 Zone2 contains all 1's
6	ECSL-ALLZERO		Shows the state of ECSL Password programmed in Flash. This is valid only when ECSL_ARMED = 1.
		0	ECSL Password of M3 Zone2 does not contain all 0's
		1	ECSL Password of M3 Zone2 contains all 0's
5	CSM-ALLZERO		Shows the state of CSM Password programmed in Flash. This is valid only when CSM_ARMED = 1.
		0	CSM Passwords of M3 Zone2 does not contain all 0's
		1	CSM Passwords of M3 Zone2 contains all 0's
4-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.8.15 Z1_GRABSECTR Register

Figure 1-140. Z1_GRABSECTR Register

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
GRABSECTB		GRABSECTC		GRABSECTD		GRABSECTE	
R-00		R-00		R-00		R-00	

15	14	13	12	11	10	9	8
GRABSECTF		GRABSECTG		GRABSECTH		GRABSECTI	
R-00		R-00		R-00		R-00	

7	6	5	4	3	2	1	0
GRABSECTJ		GRABSECTK		GRABSECTL		GRABSECTM	
R-00		R-00		R-00		R-00	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-151. Z1_GRABSECTR Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23-22	GRABSECTB		Value in this field gets loaded from Z1_GRABSECT[23:22] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector B is inaccessible
		01	Request to allocate M3 Flash Sector B to M3 Zone1
		10	Request to allocate M3 Flash Sector B to M3 Zone1
		11	Request to make M3 Flash Sector B Non-Secure
21-20	GRABSECTC		Value in this field gets loaded from Z1_GRABSECT[21:20] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector C is inaccessible.
		01	Request to allocate M3 Flash Sector C to M3 Zone1
		10	Request to allocate M3 Flash Sector C to M3 Zone1
		11	Request to make M3 Flash Sector C Non-Secure
19-18	GRABSECTD		Value in this field gets loaded from Z1_GRABSECT[19:18] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector D is inaccessible.
		01	Request to allocate M3 Flash Sector D to M3 Zone1
		10	Request to allocate M3 Flash Sector D to M3 Zone1
		11	Request to make M3 Flash Sector D Non-Secure
17-16	GRABSECTE		Value in this field gets loaded from Z1_GRABSECT[17:16] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector E is inaccessible.
		01	Request to allocate M3 Flash Sector E to M3 Zone1
		10	Request to allocate M3 Flash Sector E to M3 Zone1
		11	Request to make M3 Flash Sector E Non-Secure
15-14	GRABSECTF		Value in this field gets loaded from Z1_GRABSECT[15:14] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector F is inaccessible.
		01	Request to allocate M3 Flash Sector F to M3 Zone1
		10	Request to allocate M3 Flash Sector F to M3. Zone1
		11	Request to make M3 Flash Sector F Non-Secure

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-151. Z1_GRABSECTR Register Field Descriptions (continued)

Bit	Field	Value	Description
13-12	GRABSECTG		Value in this field gets loaded from Z1_GRABSECT[13:12] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector G is inaccessible.
		01	Request to allocate M3 Flash Sector G to M3 Zone1
		10	Request to allocate M3 Flash Sector G to M3 Zone1
		11	Request to make M3 Flash Sector G Non-Secure
11-10	GRABSECTH		Value in this field gets loaded from Z1_GRABSECT[11:10] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector H is inaccessible.
		01	Request to allocate M3 Flash Sector H to M3 Zone1
		10	Request to allocate M3 Flash Sector H to M3 Zone1
		11	Request to make M3 Flash Sector H Non-Secure. Request to make M3 Flash sector 0 Non-Secure.
9-8	GRABSECTI		Value in this field gets loaded from Z1_GRABSECT[9:8] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector I is inaccessible.
		01	Request to allocate M3 Flash Sector I to M3 Zone1
		10	Request to allocate M3 Flash Sector I to M3 Zone1
		11	Request to make M3 Flash Sector I Non-Secure
7-6	GRABSECTJ		Value in this field gets loaded from Z1_GRABSECT[7:6] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector J is inaccessible.
		01	Request to allocate M3 Flash Sector J to M3 Zone1
		10	Request to allocate M3 Flash Sector J to M3 Zone1
		11	Request to make M3 Flash Sector J Non-Secure
5-4	GRABSECTK		Value in this field gets loaded from Z1_GRABSECT[5:4] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector K is inaccessible.
		01	Request to allocate M3 Flash Sector K to M3 Zone1
		10	Request to allocate M3 Flash Sector K to M3 Zone1
		11	Request to make M3 Flash Sector K Non-Secure
3-2	GRABSECTL		Value in this field gets loaded from Z1_GRABSECT[3:2] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector L is inaccessible.
		01	Request to allocate M3 Flash Sector L to M3 Zone1
		10	Request to allocate M3 Flash Sector L to M3 Zone1
		11	Request to make M3 Flash Sector L Non-Secure
1-0	GRABSECTM		Value in this field gets loaded from Z1_GRABSECT[1:0] when a read is issued to the address location of Z1_GRABSECT in flash.
		00	Invalid – M3 Flash Sector M is inaccessible.
		01	Request to allocate M3 Flash Sector M to M3 Zone1
		10	Request to allocate M3 Flash Sector M to M3 Zone1
		11	Request to make M3 Flash Sector M Non-Secure

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.8.16 Z1_GRABRAMR Register

Figure 1-141. Z1_GRABRAMR Register

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		GRABRAM_C1		GRABRAM_C0	
R-0		R-00		R-00	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-152. Z1_GRABRAMR Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reserved
3-2	GRABRAM_C1		Value in this field gets loaded from Z1_GRABRAM[3:2] when a read is issued to the address location of Z1_ GRABRAM in flash.
		00	Invalid – M3 C1 RAM is inaccessible.
		01	Request to allocate M3 C1 RAM to M3 Zone1
		10	Request to allocate M3 C1 RAM to M3 Zone1
		11	Request to make M3 C1 RAM Non-Secure
1-0	GRABRAM_C0		Value in this field gets loaded from Z1_GRABRAM[1:0] when a read is issued to the address location of Z1_ GRABRAM in flash
		00	Invalid – M3 C0 RAM is inaccessible.
		01	Request to allocate M3 C0 RAM to M3 Zone1
		10	Request to allocate M3 C0 RAM to M3 Zone1
		11	Request to make M3 C0 RAM Non-Secure

1.13.8.17 Z2_GRABSECTR Register

Figure 1-142. Z2_GRABSECTR Register

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
GRABSECTB		GRABSECTC		GRABSECTD		GRABSECTE	
R-00		R-00		R-00		R-00	

15	14	13	12	11	10	9	8
GRABSECTF		GRABSECTG		GRABSECTH		GRABSECTI	
R-00		R-00		R-00		R-00	

7	6	5	4	3	2	1	0
GRABSECTJ		GRABSECTK		GRABSECTL		GRABSECTM	
R-00		R-00		R-00		R-00	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-153. Z2_GRABSECTR Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23-22	GRABSECTB		Value in this field gets loaded from Z2_GRABSECT[23:22] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector B is inaccessible
		01	Request to allocate M3 Flash Sector B to M3 Zone2
		10	Request to allocate M3 Flash Sector B to M3 Zone2
		11	Request to make M3 Flash Sector B Non-Secure
21-20	GRABSECTC		Value in this field gets loaded from Z2_GRABSECT[21:20] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector C is inaccessible.
		01	Request to allocate M3 Flash Sector C to M3 Zone2
		10	Request to allocate M3 Flash Sector C to M3 Zone2
		11	Request to make M3 Flash Sector C Non-Secure
19-18	GRABSECTD		Value in this field gets loaded from Z2_GRABSECT[19:18] when a read is issued to the address location of Z21_GRABSECT in flash.
		00	Invalid – M3 Flash Sector D is inaccessible.
		01	Request to allocate M3 Flash Sector D to M3 Zone2
		10	Request to allocate M3 Flash Sector D to M3 Zone2
		11	Request to make M3 Flash Sector D Non-Secure
17-16	GRABSECTE		Value in this field gets loaded from Z2_GRABSECT[17:16] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector E is inaccessible.
		01	Request to allocate M3 Flash Sector E to M3 Zone2
		10	Request to allocate M3 Flash Sector E to M3 Zone2
		11	Request to make M3 Flash Sector E Non-Secure
15-14	GRABSECTF		Value in this field gets loaded from Z2_GRABSECT[15:14] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector F is inaccessible.
		01	Request to allocate M3 Flash Sector F to M3 Zone2
		10	Request to allocate M3 Flash Sector F to M3 Zone2
		11	Request to make M3 Flash Sector F Non-Secure
13-12	GRABSECTG		Value in this field gets loaded from Z2_GRABSECT[13:12] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector G is inaccessible.
		01	Request to allocate M3 Flash Sector G to M3 Zone2
		10	Request to allocate M3 Flash Sector G to M3 Zone2
		11	Request to make M3 Flash Sector G Non-Secure
11-10	GRABSECTH		Value in this field gets loaded from Z2_GRABSECT[11:10] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector H is inaccessible.
		01	Request to allocate M3 Flash Sector H to M3 Zone2
		10	Request to allocate M3 Flash Sector H to M3 Zone2
		11	Request to make M3 Flash Sector H Non-Secure.
9-8	GRABSECTI		Value in this field gets loaded from Z2_GRABSECT[9:8] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector I is inaccessible.
		01	Request to allocate M3 Flash Sector I to M3 Zone2
		10	Request to allocate M3 Flash Sector I to M3 Zone2
		11	Request to make M3 Flash Sector I Non-Secure

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-153. Z2_GRABSECTR Register Field Descriptions (continued)

Bit	Field	Value	Description
7-6	GRABSECTJ		Value in this field gets loaded from Z2_GRABSECT[7:6] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector J is inaccessible.
		01	Request to allocate M3 Flash Sector J to M3 Zone2
		10	Request to allocate M3 Flash Sector J to M3 Zone2
		11	Request to make M3 Flash Sector J Non-Secure
5-4	GRABSECTK		Value in this field gets loaded from Z2_GRABSECT[5:4] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector K is inaccessible.
		01	Request to allocate M3 Flash Sector K to M3 Zone2
		10	Request to allocate M3 Flash Sector K to M3 Zone2
		11	Request to make M3 Flash Sector K Non-Secure
3-2	GRABSECTL		Value in this field gets loaded from Z2_GRABSECT[3:2] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector L is inaccessible.
		01	Request to allocate M3 Flash Sector L to M3 Zone2
		10	Request to allocate M3 Flash Sector L to M3 Zone2
		11	Request to make M3 Flash Sector L Non-Secure
1-0	GRABSECTM		Value in this field gets loaded from Z2_GRABSECT[1:0] when a read is issued to the address location of Z2_GRABSECT in flash.
		00	Invalid – M3 Flash Sector M is inaccessible.
		01	Request to allocate M3 Flash Sector M to M3 Zone2
		10	Request to allocate M3 Flash Sector M to M3 Zone2
		11	Request to make M3 Flash Sector M Non-Secure

1.13.8.18 Z2_GRABRAMR Register

Figure 1-143. Z2_GRABRAMR Register

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		GRABRAM_C1		GRABRAM_C0	
R-0		R-00		R-00	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-154. Z2_GRABRAMR Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reserved
3-2	GRABRAM_C1		Value in this field gets loaded from Z2_GRABRAM[3:2] when a read is issued to the address location of Z2_ GRABRAM in flash.
		00	Invalid – M3 C1 RAM is inaccessible.
		01	Request to allocate M3 C1 RAM to M3 Zone2
		10	Request to allocate M3 C1 RAM to M3 Zone2
		11	Request to make M3 C1 RAM Non-Secure

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-154. Z2_GRABRAMR Register Field Descriptions (continued)

Bit	Field	Value	Description
1-0	GRABRAM_C0		Value in this field gets loaded from Z2_GRABRAM[1:0] when a read is issued to the address location of Z2_ GRABRAM in flash
		00	Invalid – M3 C0 RAM is inaccessible.
		01	Request to allocate M3 C0 RAM to M3 Zone2
		10	Request to allocate M3 C0 RAM to M3 Zone2
		11	Request to make M3 C0 RAM Non-Secure

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.8.19 Z1_EXEONLYR Register

Figure 1-144. Z1_EXEONLYR Register

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved			EXEONLY_SECTB	EXEONLY_SECTC	EXEONLY_SECTD	EXEONLY_SECTE	EXEONLY_SECTF
R-0			R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
EXEONLY_SECTG	EXEONLY_SECTH	EXEONLY_SECTI	EXEONLY_SECTJ	EXEONLY_SECTK	EXEONLY_SECTL	EXEONLY_SECTM	EXEONLY_SECTN
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-155. Z1_EXEONLYR Register Field Descriptions

Bit	Field	Value	Description
31-13	Reserved		Reserved
12	EXEONLY_SECTB		Execute only select register sector B
		0	Execute-only is enabled for Sector B on Zone 1
		1	Execute-only is disabled for Sector B on Zone 1
11	EXEONLY_SECTC		Execute only select register sector C
		0	Execute-only is enabled for Sector C on Zone 1
		1	Execute-only is disabled for Sector C on Zone 1
10	EXEONLY_SECTD		Execute only select register sector D
		0	Execute-only is enabled for Sector D on Zone 1
		1	Execute-only is disabled for Sector D on Zone 1
9	EXEONLY_SECTE		Execute only select register sector E
		0	Execute-only is enabled for Sector E on Zone 1
		1	Execute-only is disabled for Sector E on Zone 1
8	EXEONLY_SECTF		Execute only select register sector F
		0	Execute-only is enabled forSector F on Zone 1
		1	Execute-only is disabled for Sector F on Zone 1
7	EXEONLY_SECTG		Execute only select register sector G
		0	Execute-only is enabled for Sector G on Zone 1
		1	Execute-only is disabled for Sector G on Zone 1
6	EXEONLY_SECTH		Execute only select register sector H
		0	Execute-only is enabled for Sector H on Zone 1
		1	Execute-only is disabled for Sector H on Zone 1
5	EXEONLY_SECTI		Execute only select register sector I
		0	Execute-only is enabled for Sector I on Zone 1
		1	Execute-only is disabled for Sector I on Zone 1
4	EXEONLY_SECTJ		Execute only select register sector J
		0	Execute-only is enabled for Sector J on Zone 1
		1	Execute-only disabled for Sector J on Zone 1

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-155. Z1_EXEONLYR Register Field Descriptions (continued)

Bit	Field	Value	Description
3	EXEONLY_SECTK		Execute only select register sector K
		0	Execute-only is enabled for Sector K on Zone 1
		1	Execute-only is disabled for Sector K on Zone 1
2	EXEONLY_SECTL		Execute only select register sector L
		0	Execute-only is enabled for Sector L on Zone 1
		1	Execute-only is disabled for Sector L on Zone 1
1	EXEONLY_SECTM		Execute only select register sector M
		0	Execute-only is enabled for Sector M on Zone 1
		1	Execute-only is disabled for Sector M on Zone 1
0	EXEONLY_SECTN		Execute only select register sector sector N
		0	Execute-only is enabled for Sector N on Zone 1
		1	Execute-only is disabled forSector N on Zone 1

1.13.8.20 Z2_EXEONLYR Register

Figure 1-145. Z2_EXEONLYR Register

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved			EXEONLY_SECTB	EXEONLY_SECTC	EXEONLY_SECTD	EXEONLY_SECTE	EXEONLY_SECTF
R-0			R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
EXEONLY_SECTG	EXEONLY_SECTH	EXEONLY_SECTI	EXEONLY_SECTJ	EXEONLY_SECTK	EXEONLY_SECTL	EXEONLY_SECTM	EXEONLY_SECTN
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-156. Z2_EXEONLYR Register Field Descriptions

Bit	Field	Value	Description
31-13	Reserved		Reserved
12	EXEONLY_SECTB		Execute only select register Sector B
		0	Execute-only is enabled for Sector B on Zone 2
		1	Execute-only is disabled for Sector B on Zone 2
11	EXEONLY_SECTC		Execute only select register Sector C
		0	Execute-only is enabled for Sector C on Zone 2
		1	Execute-only is disabled for Sector C on Zone 2
10	EXEONLY_SECTD		Execute only select register Sector D
		0	Execute-only is enabled for Sector D on Zone 2
		1	Execute-only is disabled for Sector D on Zone 2
9	EXEONLY_SECTE		Execute only select register Sector E
		0	Execute-only is enabled for Sector E on Zone 2
		1	Execute-only is disabled for Sector E on Zone 2

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-156. Z2_EXEONLYR Register Field Descriptions (continued)

Bit	Field	Value	Description
8	EXEONLY_SECTF		Execute only select register Sector F
		0	Execute-only is enabled for Sector F on Zone 2
		1	Execute-only is disabled for Sector F on Zone 2
7	EXEONLY_SECTG		Execute only select register Sector G
		0	Execute-only is enabled for Sector G on Zone 2
		1	Execute-only is disabled for Sector G on Zone 2
6	EXEONLY_SECTH		Execute only select register Sector H
		0	Execute-only is enabled for Sector H on Zone 2
		1	Execute-only is disabled for Sector H on Zone 2
5	EXEONLY_SECTI		Execute only select register Sector I
		0	Execute-only is enabled for Sector I on Zone 2
		1	Execute-only is disabled for Sector I on Zone 2
4	EXEONLY_SECTJ		Execute only select register Sector J
		0	Execute-only is enabled for Sector J on Zone 2
		1	Execute-only is disabled for Sector J on Zone 2
3	EXEONLY_SECTK		Execute only select register Sector K
		0	Execute-only is enabled for Sector K on Zone 2
		1	Execute-only is disabled for Sector K on Zone 2
2	EXEONLY_SECTL		Execute only select register Sector L
		0	Execute-only is enabled for Sector L on Zone 2
		1	Execute-only is disabled for Sector L on Zone 2
1	EXEONLY_SECTM		Execute only select register Sector M
		0	Execute-only is enabled for Sector M on Zone 2
		1	Execute-only is disabled for Sector M on Zone 2
0	EXEONLY_SECTN		Execute only select register Sector N
		0	Execute-only is enabled for Sector N on Zone 2
		1	Execute-only is disabled for Sector N on Zone 2

1.13.8.21 OTPSECLOCK Register

Figure 1-146. OTPSECLOCK Register

31	24	23	20	19	16
Reserved		µCRCCRCLOCK		VCUCLOCK	
R-0		R-0000		R-0000	

15	12	11	8	7	4	3	0
M3Z2PSWDLOCK		M3Z1PSWDLOCK		C28xPSWDLOCK		JTAGLOCK	
R-0000		R-0000		R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-157. OTPSECLOCK Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23-20	µCRCCRCLOCK		Value in this field gets loaded from OTPSECLOCK[23:20] when a read is issued to the OTPSECLOCK address location in OTP.
		0	µCRC does not have ability to calculate CRC on secured memories (including EXE-Only flash sectors).
		1	µCRC has the ability to calculate CRC on M3 secured memories (including EXE-Only flash sectors)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-157. OTPSECLOCK Register Field Descriptions (continued)

Bit	Field	Value	Description
19-16	VCUCLOCK		Value in this field gets loaded from OTPSECLOCK[19:16] when a read is issued to OTPSECLOCK address location in OTP.
		0	VCU does not have the ability to calculate CRC on secured memories (including EXE-Only flash sectors).
		1	VCU has the ability to calculate CRC on C28x secured memories (including EXE-Only flash sectors)
15-12	M3Z2PSWDLOCK		Value in this field gets loaded from OTPSECLOCK[15:12] when a read is issued to OTPSECLOCK address location in OTP.
		0	M3 Zone2 CSM/ECSL passwords can be accessed from the debugger as well as code running from any memory.
		1	M3 Zone2 CSM/ECSL passwords cannot be accessed from the debugger as well as code running from unsecure memory. It can be read from code running from secure memory, which belongs to M3 Zone2.
11-8	M3Z1PSWDLOCK		Value in this field gets loaded from OTPSECLOCK[11:8] when a read is issued to the OTPSECLOCK address location in OTP.
		0	M3 Zone1 CSM/ECSL passwords can be accessed from the debugger as well as code running from any memory.
		1	M3 Zone1 CSM/ECSL passwords cannot be accessed from the debugger as well as code running from unsecure memory. It can be read from code running from secure memory, which belongs to M3 Zone1
7-4	C28xPSWDLOCK		Value in this field gets loaded from OTPSECLOCK[7:4] when a read is issued to the OTPSECLOCK address location in OTP.
		0	C28x Zone1 CSM/ECSL passwords can be accessed from the debugger as well as code running from any memory.
		1	C28x Zone1 CSM/ECSL passwords cannot be accessed from the debugger as well as code running from unsecure memory. It can be read from code running from secure memory, which belongs to C28x Zone1.
3-0	JTAGLOCK		Value in this field gets loaded from OTPSECLOCK[3:0] when a read is issued to the OTPSECLOCK address location in OTP.
		0	JTAG Port is disabled. Debugger (for example, CCS) cannot be connected to the Cortex-M3 as well as the C28x.
		1	JTAG Port is enabled. Debugger (for example, CCS) can be connected to the Cortex-M3 as well as the C28x.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.9 Control Subsystem Code Security Module (CSM) Registers

1.13.9.1 CSMKEY0 Register

Figure 1-147. Z1_CSMKEY0 Register

31	0
CSMKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-158. CSMKEY0 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unclock the control subsystem, the user needs to write the same value in CSMPSWD0 to this register.

1.13.9.2 CSMKEY1 Register

Figure 1-148. CSMKEY1 Register

31	0
CSMKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-159. CSMKEY1 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unclock the control subsystem, the user needs to write the same value in CSMPSWD1 to this register.

1.13.9.3 CSMKEY2 Register

Figure 1-149. CSMKEY2 Register

31	0
CSMKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-160. CSMKEY2 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unclock the control subsystem, the user needs to write the same value in CSMPSWD2 to this register.

1.13.9.4 CSMKEY3 Register

Figure 1-150. CSMKEY3 Register

31	0
CSMKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-161. CSMKEY3 Register Field Descriptions

Bit	Field	Value	Description
31-0	CSMKEY		To unclock the control subsystem, the user needs to write the same value in CSMPSWD3 to this register.

1.13.9.5 CSMCR Register

Figure 1-151. CSMCR Register

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
FORCESEC	Reserved				ECSL-ARMED	ECSL-MATCH	ECSL-ALLONE
R=0/W=1	R-0				R-x	R-x	R-x

7	6	5	4	3	2	1	0
ECSL-ALLONE	CSM-ARMED	CSM-MATCH	CSM_ALLONE	CSM_ALLZERO		Reserved	
R-x	R-x	R-x	R-x	R-x		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-162. CSMCR Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	FORCESEC		Writing a 1 to this bit clears all the KEY (CSM & ECSL both) registers for the control subsystem and makes zone1 secure if CSMPSWD are not all 1's.
		0	Read always return 0 (Self clear bit).
		1	Clears the KEY registers and makes the control subsystem secure. The password match flow described in Section 1.10.3.2 must be followed to unsecure zone1 again.
14-11	Reserved		Reserved
10	ECSL-ARMED	0	Dummy read to the control subsystem ECSL PWL (Password locations) have not been performed.
		1	Dummy read to the control subsystem ECSL PWL (Password locations) have been performed.
9	ECSL_MATCH		Status bit indicate if ECSL is enabled or disabled for the control subsystem. This is valid only when ECSL_ARMED = 1.
		0	Control subsystem ECSL is enabled.
		1	Control subsystem ECSL is disabled.
8	ECSL_ALLONE		Shows the state of ECSL Password programmed in Flash. This is valid only when ECSL _ARMED = 1.
		0	ECSL Password of control subsystem does not contain all 1's
		1	ECSL Password of control subsystem contain all 1's
7	ECSL_ALLZERO		Shows the state of ECSL password programmed in flash. This is valid only when ECSL_ARMED = 1.
		0	ECSL Password of control subsystem does not contain all 0's
		1	ECSL Password of control subsystem contains all 0's
6	CSM_ARMED	0	Dummy read to control subsystem CSM PWL (Password locations) have not been performed.
		1	Dummy read to control subsystem CSM PWL (Password locations) have been performed.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-162. CSMCR Register Field Descriptions (continued)

Bit	Field	Value	Description
5	CSM_MATCH		Status bit that reflects the security state of the control subsystem. This is valid only when CSM_ARMED = 1.
		0	Control subsystem is secure if CSM-ALLONE is 0.
		1	Control subsystem is non-secure if CSM-ALLZERO is 0.
4	CSM_ALLONE		Shows the state of CSM password programmed in flash. This is valid only when CSM_ARMED = 1.
		0	CSM Passwords of control subsystem does not contain all 1's
		1	CSM Passwords of control subsystem contains all 1's
3	CSM_ALLZERO		Shows the state of CSM Password programmed in Flash. This is valid only when CSM_ARMED = 1.
		0	CSM Passwords of control subsystem does not contain all 0's
		1	CSM Passwords of control subsystem contains all 0's
2-0	Reserved		Reserved

1.13.9.6 ECSLKEY0 Register

Figure 1-152. ECSLKEY0 Register

31	0
ECSLKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-163. ECSLKEY0 Register Field Descriptions

Bit	Field	Value	Description
31-0	ECSLKEY		To disable ECSL logic active on the control subsystem, the user needs to write the same value in ECSLPSWD0 of zone1 to this register.

1.13.9.7 ECSLKEY1 Register

Figure 1-153. ECSLKEY1 Register

31	0
ECSLKEY	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-164. ECSLKEY1 Register Field Descriptions

Bit	Field	Value	Description
31-0	ECSLKEY		To disable ECSL logic active on the control subsystem, the user needs to write the same value in ECSLPSWD1 of zone1 to this register.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.9.8 EXEONLYR Register

Figure 1-154. EXEONLYR Register

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved		EXEONLY-SECTA	EXEONLY_SECTB	EXEONLY_SECTC	EXEONLY_SECTD	EXEONLY_SECTE	EXEONLY_SECTF
R-0		R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
EXEONLY_SECTG	EXEONLY_SECTH	EXEONLY_SECTI	EXEONLY_SECTJ	EXEONLY_SECTK	EXEONLY_SECTL	EXEONLY_SECTM	EXEONLY_SECTN
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-165. EXEONLYR Register Field Description

Bit	Field	Value	Description
31-14	Reserved		Reserved
13	EXEONLY-SECTA		Value in this field gets loaded from EXEONLY[13:13] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x flash sector A.
		1	Execute-Only protection is disabled for C28x flash sector A.
12	EXEONLY_SECTB		Value in this field gets loaded from EXEONLY[12:12] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector B (only if it is allocated to the control subsystem)
		1	Execute-Only protection is disabled for C28x Flash Sector B (only if it is allocated to the control subsystem)
11	EXEONLY_SECTC		Value in this field gets loaded from EXEONLY[11:11] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector C (only if it is allocated to the control subsystem)
		1	Execute-Only protection is disabled for C28x Flash Sector C (only if it is allocated to the control subsystem)
10	EXEONLY_SECTD		Value in this field gets loaded from EXEONLY[10:10] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector D (only if it is allocated to the control subsystem)
		1	Execute-Only protection is disabled for C28x Flash Sector D (only if it is allocated to the control subsystem)
9	EXEONLY_SECTE		Value in this field gets loaded from EXEONLY[9:9] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector E (only if it is allocated to C28x zone1)
		1	Execute-Only protection is disabled for C28x Flash Sector E (only if it is allocated to C28x zone1)
8	EXEONLY_SECTF		Value in this field gets loaded from EXEONLY[8:8] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector F (only if it is allocated to the control subsystem)
		1	Execute-Only protection is disabled for C28x Flash Sector F (only if it is allocated to the control subsystem)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-165. EXEONLYR Register Field Description (continued)

Bit	Field	Value	Description
7	EXEONLY_SECTG		Value in this field gets loaded from EXEONLY[7:7] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector G (only if it is allocated to the control subsystem)
		1	Execute-Only protection is disabled for C28x Flash Sector G (only if it is allocated to the control subsystem)
6	EXEONLY_SECTH		Value in this field gets loaded from EXEONLY[6:6] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector H (only if it is allocated to the control subsystem)
		1	Execute-Only protection is disabled for C28x Flash Sector H (only if it is allocated to the control subsystem)
5	EXEONLY_SECTI		Value in this field gets loaded from EXEONLY[5:5] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector I (only if it is allocated to the control subsystem)
		1	Execute-Only protection is disabled for C28x Flash Sector I (only if it is allocated to the control subsystem)
4	EXEONLY_SECTJ		Value in this field gets loaded from EXEONLY[4:4] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector J (only if it is allocated to the control subsystem)
		1	Execute-Only protection is disabled for C28x Flash Sector J (only if it is allocated to the control subsystem)
3	EXEONLY_SECTK		Value in this field gets loaded from EXEONLY[3:3] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector K (only if it is allocated to C28x zone1)
		1	Execute-Only protection is disabled for C28x Flash Sector K (only if it is allocated to C28x zone1)
2	EXEONLY_SECTL		Value in this field gets loaded from EXEONLY[2:2] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector L (only if it is allocated to the control subsystem)
		1	Execute-Only protection is disabled for C28x Flash Sector L (only if it is allocated to the control subsystem)
1	EXEONLY_SECTM		Value in this field gets loaded from EXEONLY[1:1] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector M (only if it is allocated to the control subsystem)
		1	Execute-Only protection is disabled for C28x Flash Sector M (only if it is allocated to the control subsystem)
0	EXEONLY_SECTN		Value in this field gets loaded from EXEONLY[0:0] when a read is issued to the EXEONLY address location in flash.
		0	Execute-Only protection is enabled for C28x Flash Sector N (Dedicated for the control subsystem only)
		1	Execute-Only protection is disabled for C28x Flash Sector N (Dedicated for the control subsystem only)

1.13.10 µCRC Register Description

The following table summarizes the registers for µCRC.

Table 1-166. µCRC Register Summary

Name	Address (Offset)	Size (X32)	Description
µCRCCONFIG	0x0	1	µCRC Configuration Register
µCRCCONTROL	0x4	1	µCRC Control Register

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-166. µCRC Register Summary (continued)

Name	Address (Offset)	Size (X32)	Description
µCRCRES	0x8	1	µCRC Result Register

1.13.10.1 µCRCCONFIG Register

Figure 1-155. µCRCCONFIG Register

31	2	1	0
Reserved			CRCTYPE
R-0			R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-167. µCRCCONFIG Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
30	CRCTYPE		These bits configure the polynomial used for CRC calculation.
		00	CRC8 (Polynomial = 0x07)
		01	CRC16-P1 (Polynomial = 0x8005)
		10	CRC16-P2 (Polynomial = 0x1021)
		11	CRC32 (Polynomial = 0x04C11DB7)

1.13.10.2 µCRCCONTROL Register

Figure 1-156. µCRCCONTROL Register

31	2	1	0
Reserved		SOFTRESET	CLEAR
R-0		R-0/W-1	R-0/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-168. µCRCCONTROL Register Field Descriptions

Bit	Field	Value	Description
31-2			Reserved
1	SOFTRESET		This is the auto clear field. Reads to this bit always return ‘0’
		0	No effect.
		1	Issues a reset to the uCRC module
0	CLEAR		This is the auto clear field. Reads to this bit always return ‘0’.
		0	No effect
		1	Clears the content in uCRCRES register.

1.13.10.3 µCRCRES Register

Figure 1-157. µCRCRES Register

31	0
RESULT	
	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-169. µCRCRES Register Field Descriptions

Bit	Field	Value	Description
31-0	RESULT		This register contains the calculated CRC. This gets updated for every read (byte access) to mirrored address locations.

1.13.11 Master Subsystem IPC Registers

The below registers are mapped to the master subsystem address map only.

1.13.11.1 M3 to C28 IPC Set (MTOCIPCSET) Register

Figure 1-158. M3 to C28 IPC Set (MTOCIPCSET) Register

31	30	29	28	27	26	25	24
IPC32	IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IPC16	IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-170. M3 to C28 IPC Set (MTOCIPCSET) Field Descriptions

Bit	Field	Value	Description
31	IPC32	0	MTOCIPCSET Flag 32. M3 to C28 core IPC flag 32 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
30	IPC31	0	MTOCIPCSET Flag 31. M3 to C28 core IPC flag 31 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
29	IPC30	0	MTOCIPCSET Flag 30. M3 to C28 core IPC flag 30 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
28	IPC29	0	MTOCIPCSET Flag 29. M3 to C28 core IPC flag 29 set. If a bit is set by writing a ‘‘1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
27	IPC28	0	MTOCIPCSET Flag 28. M3 to C28 core IPC flag 28 set. If a bit is set by writing a ‘‘1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
26	IPC27	0	MTOCIPCSET Flag 27. M3 to C28 core IPC flag 27 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
25	IPC26	0	MTOCIPCSET Flag 26. M3 to C28 core IPC flag 26 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
24	IPC25	0	MTOCIPCSET Flag 25. M3 to C28 core IPC flag 25 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
23	IPC24	0	MTOCIPCSET Flag 24. M3 to C28 core IPC flag 24 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-170. M3 to C28 IPC Set (MTOCIPCSET) Field Descriptions (continued)

Bit	Field	Value	Description
22	IPC23	0	MTOCIPCSET Flag 23. M3 to C28 core IPC flag 23 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
21	IPC22	0	MTOCIPCSET Flag 22. M3 to C28 core IPC flag 22 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
20	IPC21	0	MTOCIPCSET Flag 21. M3 to C28 core IPC flag 21 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
19	IPC20	0	MTOCIPCSET Flag 20. M3 to C28 core IPC flag 20 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
18	IPC19	0	MTOCIPCSET Flag 19. M3 to C28 core IPC flag 19 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
17	IPC18	0	MTOCIPCSET Flag 18. M3 to C28 core IPC flag 18 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
16	IPC17	0	MTOCIPCSET Flag 17. M3 to C28 core IPC flag 17 set. If a bit is set by writing a ‘‘1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
15	IPC16	0	MTOCIPCSET Flag 16. M3 to C28 core IPC flag 16 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
14	IPC15	0	MTOCIPCSET Flag 15. M3 to C28 core IPC flag 15 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
13	IPC14	0	MTOCIPCSET Flag 14. M3 to C28 core IPC flag 14 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
12	IPC13	0	MTOCIPCSET Flag 13. M3 to C28 core IPC flag 13 set. If a bit is set by writing a ‘‘1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
11	IPC12	0	MTOCIPCSET Flag 12. M3 to C28 core IPC flag 12 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
10	IPC11	0	MTOCIPCSET Flag 11. M3 to C28 core IPC flag 11 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
9	IPC10	0	MTOCIPCSET Flag 10. M3 to C28 core IPC flag 10 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
8	IPC9	0	MTOCIPCSET Flag 9. M3 to C28 core IPC flag 9 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
7	IPC8	0	MTOCIPCSET Flag 8. M3 to C28 core IPC flag 8 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
6	IPC7	0	MTOCIPCSET Flag 7. M3 to C28 core IPC flag 7 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
5	IPC6	0	MTOCIPCSET Flag 6. M3 to C28 core IPC flag 6 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
4	IPC5	0	MTOCIPCSET Flag 5. M3 to C28 core IPC flag 5 set. If a bit is set by writing a '1' then the corresponding bit in MTOCIPCFLG is set. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
3	IPC4	0	MTOCIPCSET Interrupt 4. M3 to C28 IPC interrupt 4 request set. If this bit is set by writing a '1' then MTOCINT1 is raised to the C28 PIE. The status of this bit is not readable in this register – it is readable in the corresponding bit in the MTOCIPCFLG and STS registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-170. M3 to C28 IPC Set (MTOCIPCSET) Field Descriptions (continued)

Bit	Field	Value	Description
2	IPC3	0	MTOCIPCSET Interrupt 3. M3 to C28 IPC interrupt 3 request set. If this bit is set by writing a '1' then MTOCINT1 is raised to the C28 PIE. The status of this bit is not readable in this register – it is readable in the corresponding bit in the MTOCIPCFLG and STS registers.
1	IPC2	0	MTOCIPCSET Interrupt 2. M3 to C28 IPC interrupt 2 request set. If this bit is set by writing a '1' then MTOCINT1 is raised to the C28 PIE. The status of this bit is not readable in this register – it is readable in the corresponding bit in the MTOCIPCFLG and STS registers.
0	IPC1	0	MTOCIPCSET Interrupt 1. M3 to C28 IPC interrupt 1 request set. If this bit is set by writing a '1' then MTOCINT1 is raised to the C28 PIE. The status of this bit is not readable in this register – it is readable in the corresponding bit in the MTOCIPCFLG and STS registers.

1.13.11.2 M3 to C28 IPC Clear (MTOCIPCCLR) Register

Figure 1-159. M3 to C28 IPC Clear (MTOCIPCCLR) Register

31	30	29	28	27	26	25	24
IPC32	IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IPC16	IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-171. M3 to C28 IPC Clear (MTOCIPCCLR) Register Field Descriptions

Bit	Field	Value	Description
31	IPC32	0	MTOCIPCCLR Flag 32. M3 to C28 core IPC flag 32 clear. If a bit is cleared by writing a ‘1’ then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
30	IPC31	0	MTOCIPCCLR Flag 31. M3 to C28 core IPC flag 31 clear. If a bit is cleared by writing a ‘1’ then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
29	IPC30	0	MTOCIPCCLR Flag 30. M3 to C28 core IPC flag 30 clear. If a bit is cleared by writing a ‘1’ then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
28	IPC29	0	MTOCIPCCLR Flag 29. M3 to C28 core IPC flag 29 clear. If a bit is cleared by writing a ‘1’ then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
27	IPC28	0	MTOCIPCCLR Flag 28. M3 to C28 core IPC flag 28 clear. If a bit is cleared by writing a ‘1’ then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
26	IPC27	0	MTOCIPCCLR Flag 27. M3 to C28 core IPC flag 27 clear. If a bit is cleared by writing a ‘1’ then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
25	IPC26	0	MTOCIPCCLR Flag 26. M3 to C28 core IPC flag 26 clear. If a bit is cleared by writing a ‘1’ then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
24	IPC25	0	MTOCIPCCLR Flag 25. M3 to C28 core IPC flag 25 clear. If a bit is cleared by writing a ‘1’ then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-171. M3 to C28 IPC Clear (MTOCIPCCLR) Register Field Descriptions (continued)

Bit	Field	Value	Description
23	IPC24	0	MTOCIPCCLR Flag 24. M3 to C28 core IPC flag 24 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
22	IPC23	0	MTOCIPCCLR Flag 23. M3 to C28 core IPC flag 23 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
21	IPC22	0	MTOCIPCCLR Flag 22. M3 to C28 core IPC flag 22 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
20	IPC21	0	MTOCIPCCLR Flag 21. M3 to C28 core IPC flag 21 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
19	IPC20	0	MTOCIPCCLR Flag 20. M3 to C28 core IPC flag 20 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
18	IPC19	0	MTOCIPCCLR Flag 19. M3 to C28 core IPC flag 19 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
17	IPC18	0	MTOCIPCCLR Flag 18. M3 to C28 core IPC flag 18 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
16	IPC17	0	MTOCIPCCLR Flag 17. M3 to C28 core IPC flag 17 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
15	IPC16	0	MTOCIPCCLR Flag 16. M3 to C28 core IPC flag 16 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
14	IPC15	0	MTOCIPCCLR Flag 15. M3 to C28 core IPC flag 15 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
13	IPC14	0	MTOCIPCCLR Flag 14. M3 to C28 core IPC flag 14 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
12	IPC13	0	MTOCIPCCLR Flag 13. M3 to C28 core IPC flag 13 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
11	IPC12	0	MTOCIPCCLR Flag 12. M3 to C28 core IPC flag 12 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
10	IPC11	0	MTOCIPCCLR Flag 11. M3 to C28 core IPC flag 11 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
9	IPC10	0	MTOCIPCCLR Flag 10. M3 to C28 core IPC flag 10 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
8	IPC9	0	MTOCIPCCLR Flag 9. M3 to C28 core IPC flag 9 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
7	IPC8	0	MTOCIPCCLR Flag 8. M3 to C28 core IPC flag 8 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
6	IPC7	0	MTOCIPCCLR Flag 7. M3 to C28 core IPC flag 8 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
5	IPC6	0	MTOCIPCCLR Flag 6. M3 to C28 core IPC flag 6 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
4	IPC5	0	MTOCIPCCLR Flag 5. M3 to C28 core IPC flag 5 clear. If a bit is cleared by writing a '1' then the corresponding bit in MTOCIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-171. M3 to C28 IPC Clear (MTOCIPCCLR) Register Field Descriptions (continued)

Bit	Field	Value	Description
3	IPC4	0	MTOCIPCCLR Interrupt 4. M3 to C28 IPC interrupt 4 request clear. If this bit is cleared by writing a '1' then MTOCINT1 is raised to the C28 PIE. The status of this bit is not readable in this register – it is readable in the corresponding bit in the MTOCIPCFLG and STS registers.
2	IPC3	0	MTOCIPCCLR Interrupt 3. M3 to C28 IPC interrupt 3 request clear. If this bit is cleared by writing a '1' then MTOCINT1 is raised to the C28 PIE. The status of this bit is not readable in this register – it is readable in the corresponding bit in the MTOCIPCFLG and STS registers.
1	IPC2	0	MTOCIPCCLR Interrupt 2. M3 to C28 IPC interrupt 2 request clear. If this bit is cleared by writing a '1' then MTOCINT1 is raised to the C28 PIE. The status of this bit is not readable in this register – it is readable in the corresponding bit in the MTOCIPCFLG and STS registers.
0	IPC1	0	MTOCIPCCLR Interrupt 1. M3 to C28 IPC interrupt 1 request clear. If this bit is cleared by writing a '1' then MTOCINT1 is raised to the C28 PIE. The status of this bit is not readable in this register – it is readable in the corresponding bit in the MTOCIPCFLG and STS registers..

1.13.11.3 M3 to C28 Core Flag (MTOCIPCFLG) Register

Figure 1-160. M3 to C28 Core Flag (MTOCIPCFLG) Register

31	30	29	28	27	26	25	24
IPC32	IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
IPC16	IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-172. M3 to C28 Core Flag (MTOCIPCFLG) Register Field Descriptions

Bit	Field	Value	Description
31	IPC32	0	MTOCIPCFLG Flag 32. M3 to C28 core IPC flag 32 status. The bit is '1' if the corresponding MTOCIPCSET bit has been written with a '1' and MTOCIPCCLR or MTOCIPCACK bit has not been written with a '1."
30	IPC31	0	MTOCIPCFLG Flag 31. M3 to C28 core IPC flag 31 status. The bit is '1' if the corresponding MTOCIPCSET bit has been written with a '1' and MTOCIPCCLR or MTOCIPCACK bit has not been written with a '1."
29	IPC30	0	MTOCIPCFLG Flag 30. M3 to C28 core IPC flag 30 status. The bit is '1' if the corresponding MTOCIPCSET bit has been written with a '1' and MTOCIPCCLR or MTOCIPCACK bit has not been written with a '1."
28	IPC29	0	MTOCIPCFLG Flag 29. M3 to C28 core IPC flag 29 status. The bit is '1' if the corresponding MTOCIPCSET bit has been written with a '1' and MTOCIPCCLR or MTOCIPCACK bit has not been written with a '1."
27	IPC28	0	MTOCIPCFLG Flag 28. M3 to C28 core IPC flag 28 status. The bit is '1' if the corresponding MTOCIPCSET bit has been written with a '1' and MTOCIPCCLR or MTOCIPCACK bit has not been written with a '1."
26	IPC27	0	MTOCIPCFLG Flag 27. M3 to C28 core IPC flag 27 status. The bit is '1' if the corresponding MTOCIPCSET bit has been written with a '1' and MTOCIPCCLR or MTOCIPCACK bit has not been written with a '1."
25	IPC26	0	MTOCIPCFLG Flag 26..M3 to C28 core IPC flag 26 status. The bit is '1' if the corresponding MTOCIPCSET bit has been written with a '1' and MTOCIPCCLR or MTOCIPCACK bit has not been written with a '1."

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-172. M3 to C28 Core Flag (MTOCIPCFLG) Register Field Descriptions (continued)

Bit	Field	Value	Description
24	IPC25	0	MTOCIPCFLG Flag 25. M3 to C28 core IPC flag 25 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
23	IPC24	0	MTOCIPCFLG Flag 24. M3 to C28 core IPC flag 24 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
22	IPC23	0	MTOCIPCFLG Flag 23. M3 to C28 core IPC flag 22 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
21	IPC22	0	MTOCIPCFLG Flag 22. M3 to C28 core IPC flag 22 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
20	IPC21	0	MTOCIPCFLG Flag 21. M3 to C28 core IPC flag 21 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
19	IPC20	0	MTOCIPCFLG Flag 20. M3 to C28 core IPC flag 20 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’"
18	IPC19	0	MTOCIPCFLG Flag 19. M3 to C28 core IPC flag 19 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
17	IPC18	0	MTOCIPCFLG Flag 18. M3 to C28 core IPC flag 18 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
16	IPC17	0	MTOCIPCFLG Flag 17. M3 to C28 core IPC flag 17 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
15	IPC16	0	MTOCIPCFLG Flag 16. M3 to C28 core IPC flag 16 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
14	IPC15	0	MTOCIPCFLG Flag 15. M3 to C28 core IPC flag 15 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
13	IPC14	0	MTOCIPCFLG Flag 14. M3 to C28 core IPC flag 14 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
12	IPC13	0	MTOCIPCFLG Flag 13. M3 to C28 core IPC flag 13 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
11	IPC12	0	MTOCIPCFLG Flag 12. M3 to C28 core IPC flag 12 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
10	IPC11	0	MTOCIPCFLG Flag 11. M3 to C28 core IPC flag 11 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
9	IPC10	0	MTOCIPCFLG Flag 10. M3 to C28 core IPC flag 10 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
8	IPC9	0	MTOCIPCFLG Flag 9. M3 to C28 core IPC flag 9 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
7	IPC8	0	MTOCIPCFLG Flag 8. M3 to C28 core IPC flag 8 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
6	IPC7	0	MTOCIPCFLG Flag 7. M3 to C28 core IPC flag 7 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
5	IPC6	0	MTOCIPCFLG Flag 6. M3 to C28 core IPC flag 6 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-172. M3 to C28 Core Flag (MTOCIPCFLG) Register Field Descriptions (continued)

Bit	Field	Value	Description
4	IPC5	0	MTOCIPCFLG Flag 5. M3 to C28 core IPC flag 5 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
3	IPC4	0	MTOCIPCFLG Interrupt 4. M3 to C28 IPC interrupt 4 status flag. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
2	IPC3	0	MTOCIPCFLG Interrupt 3. M3 to C28 IPC interrupt 3 status flag. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
1	IPC2	0	MTOCIPCFLG Interrupt 2. M3 to C28 IPC interrupt 2 status flag. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."
0	IPC1	0	MTOCIPCFLG Interrupt 1. M3 to C28 IPC interrupt 1 status flag. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1."

1.13.11.4 C28 to M3 Core IPC Acknowledge (CTOMIPCACK) Register

Figure 1-161. M3 to C28 Core IPC Acknowledge (CTOMIPCACK) Register

31	30	29	28	27	26	25	24
IPC32	IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IPC16	IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-173. M3 to C28 Core IPC Acknowledge (CTOMIPCACK) Register Field Descriptions

Bit	Field	Value	Description
31	IPC32	0	CTOMIPCACK Flag 32. C28 to M3 core IPC flag 32 acknowledge. Writing a ‘1’ to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to ‘0’. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
30	IPC31	0	CTOMIPCACK Flag 31. C28 to M3 core IPC flag 31 acknowledge. Writing a ‘1’ to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to ‘0’. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
29	IPC30	0	CTOMIPCACK Flag 30. C28 to M3 core IPC flag 30 acknowledge. Writing a ‘1’ to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to ‘0’. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
28	IPC29	0	CTOMIPCACK Flag 29. C28 to M3 core IPC flag 29 acknowledge. Writing a ‘1’ to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to ‘0’. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers
27	IPC28	0	CTOMIPCACK Flag 28. C28 to M3 core IPC flag 28 acknowledge. Writing a ‘1’ to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to ‘0’. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
26	IPC27	0	CTOMIPCACK Flag 27. C28 to M3 core IPC flag 27 acknowledge. Writing a ‘1’ to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to ‘0’. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-173. M3 to C28 Core IPC Acknowledge (CTOMIPCACK) Register Field Descriptions (continued)

Bit	Field	Value	Description
25	IPC26	0	CTOMIPCACK Flag 26. C28 to M3 core IPC flag 26 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
24	IPC25	0	CTOMIPCACK Flag 25. C28 to M3 core IPC flag 25 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers
23	IPC24	0	CTOMIPCACK Flag 24. C28 to M3 core IPC flag 24 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
22	IPC23	0	CTOMIPCACK Flag 23. C28 to M3 core IPC flag 23 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
21	IPC22	0	CTOMIPCACK Flag 22. C28 to M3 core IPC flag 22 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
20	IPC21	0	CTOMIPCACK Flag 21. C28 to M3 core IPC flag 21 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
19	IPC20	0	CTOMIPCACK Flag 20. C28 to M3 core IPC flag 20 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
18	IPC19	0	CTOMIPCACK Flag 19. C28 to M3 core IPC flag 19 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
17	IPC18	0	CTOMIPCACK Flag 18. C28 to M3 core IPC flag 18 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
16	IPC17	0	CTOMIPCACK Flag 17. C28 to M3 core IPC flag 17 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
15	IPC16	0	CTOMIPCACK Flag 16. C28 to M3 core IPC flag 16 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
14	IPC15	0	CTOMIPCACK Flag 15. C28 to M3 core IPC flag 15 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
13	IPC14	0	CTOMIPCACK Flag 14. C28 to M3 core IPC flag 14 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
12	IPC13	0	CTOMIPCACK Flag 13. C28 to M3 core IPC flag 13 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
11	IPC12	0	CTOMIPCACK Flag 12. C28 to M3 core IPC flag 12 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
10	IPC11	0	CTOMIPCACK Flag 11. C28 to M3 core IPC flag 11 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
9	IPC10	0	CTOMIPCACK Flag 10. C28 to M3 core IPC flag 10 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
8	IPC9	0	CTOMIPCACK Flag 9. C28 to M3 core IPC flag 9 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
7	IPC8	0	CTOMIPCACK Flag 8. C28 to M3 core IPC flag 8 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
6	IPC7	0	CTOMIPCACK Flag 7. C28 to M3 core IPC flag 7 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-173. M3 to C28 Core IPC Acknowledge (CTOMIPCACK) Register Field Descriptions (continued)

Bit	Field	Value	Description
5	IPC6	0	CTOMIPCACK Flag 6. C28 to M3 core IPC flag 6 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
4	IPC5	0	CTOMIPCACK Flag 5. C28 to M3 core IPC flag 5 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
3	IPC4	0	CTOMIPCACK Interrupt 4. C28 to M3 IPC interrupt 4 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
2	IPC3	0	CTOMIPCACK Interrupt 3. C28 to M3 IPC interrupt 3 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
1	IPC2	0	CTOMIPCACK Interrupt 2. C28 to M3 IPC interrupt 2 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
0	IPC1	0	CTOMIPCACK Interrupt 1. C28 to M3 IPC interrupt 1 acknowledge. Writing a '1' to this bit clears the corresponding bit in CTOMIPCFLG and CTOMIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.

1.13.11.5 C28 to M3 Core IPC Status (CTOMIPCSTS) Register

Figure 1-162. C28 to M3 Core IPC Status (CTOMIPCSTS) Register

31	30	29	28	27	26	25	24
IPC32	IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
IPC16	IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 1-174. C28 to M3 Core IPC Status (CTOMIPCSTS) Register Field Descriptions

Bit	Field	Value	Description
31	IPC32	0	CTOMIPCSTS Flag 32. C28 to M3 core IPC flag 32 status. The bit is '1' if the corresponding CTOMIPCSET bit has been written with a '1' and CTOMIPCCLR or CTOMIPCACK bit has not been written with a '1."
30	IPC31	0	CTOMIPCSTS Flag 31. C28 to M3 core IPC flag 31 status. The bit is '1' if the corresponding CTOMIPCSET bit has been written with a '1' and CTOMIPCCLR or CTOMIPCACK bit has not been written with a '1."
29	IPC30	0	CTOMIPCSTS Flag 30. C28 to M3 core IPC flag 30 status. The bit is '1' if the corresponding CTOMIPCSET bit has been written with a '1' and CTOMIPCCLR or CTOMIPCACK bit has not been written with a '1."
28	IPC29	0	CTOMIPCSTS Flag 29. C28 to M3 core IPC flag 29 status. The bit is '1' if the corresponding CTOMIPCSET bit has been written with a '1' and CTOMIPCCLR or CTOMIPCACK bit has not been written with a '1."
27	IPC28	0	CTOMIPCSTS Flag 28. C28 to M3 core IPC flag 28 status. The bit is '1' if the corresponding CTOMIPCSET bit has been written with a '1' and CTOMIPCCLR or CTOMIPCACK bit has not been written with a '1."

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-174. C28 to M3 Core IPC Status (CTOMIPCSTS) Register Field Descriptions (continued)

Bit	Field	Value	Description
26	IPC27	0	CTOMIPCSTS Flag 27. C28 to M3 core IPC flag 27 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
25	IPC26	0	CTOMIPCSTS Flag 26. C28 to M3 core IPC flag 26 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
24	IPC25	0	CTOMIPCSTS Flag 25. C28 to M3 core IPC flag 25 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
23	IPC24	0	CTOMIPCSTS Flag 24. C28 to M3 core IPC flag 24 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
22	IPC23	0	CTOMIPCSTS Flag 23. C28 to M3 core IPC flag 23 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
21	IPC22	0	CTOMIPCSTS Flag 22. C28 to M3 core IPC flag 22 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
20	IPC21	0	CTOMIPCSTS Flag 21. C28 to M3 core IPC flag 21 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
19	IPC20	0	CTOMIPCSTS Flag 20. C28 to M3 core IPC flag 20 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
18	IPC19	0	CTOMIPCSTS Flag 19. C28 to M3 core IPC flag 19 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
17	IPC18	0	CTOMIPCSTS Flag 18. C28 to M3 core IPC flag 18 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
16	IPC17	0	CTOMIPCSTS Flag 17. C28 to M3 core IPC flag 17 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
15	IPC16	0	CTOMIPCSTS Flag 16. C28 to M3 core IPC flag 16 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
14	IPC15	0	CTOMIPCSTS Flag 15. C28 to M3 core IPC flag 15 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
13	IPC14	0	CTOMIPCSTS Flag 14. C28 to M3 core IPC flag 14 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
12	IPC13	0	CTOMIPCSTS Flag 13. C28 to M3 core IPC flag 13 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
11	IPC12	0	CTOMIPCSTS Flag 12. C28 to M3 core IPC flag 12 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
10	IPC11	0	CTOMIPCSTS Flag 11. C28 to M3 core IPC flag 11 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
9	IPC10	0	CTOMIPCSTS Flag 10. C28 to M3 core IPC flag 10 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
8	IPC9	0	CTOMIPCSTS Flag 9. C28 to M3 core IPC flag 9 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
7	IPC8	0	CTOMIPCSTS Flag 8. C28 to M3 core IPC flag 8 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-174. C28 to M3 Core IPC Status (CTOMIPCSTS) Register Field Descriptions (continued)

Bit	Field	Value	Description
6	IPC7	0	CTOMIPCSTS Flag 7. C28 to M3 core IPC flag 7 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
5	IPC6	0	CTOMIPCSTS Flag 6. C28 to M3 core IPC flag 6 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
4	IPC5	0	CTOMIPCSTS Flag 5. C28 to M3 core IPC flag 5 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
3	IPC4	0	CTOMIPCSTS Interrupt 4. C28 to M3 IPC interrupt 4 status flag. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
2	IPC3	0	CTOMIPCSTS Interrupt 3. C28 to M3 IPC interrupt 3 status flag. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
1	IPC2	0	CTOMIPCSTS Interrupt 2. C28 to M3 IPC interrupt 2 status flag. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."
0	IPC1	0	CTOMIPCSTS Interrupt 1. C28 to M3 IPC interrupt 1 status flag. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1."

1.13.11.6 M3 Flash Semaphore Register

Figure 1-163. M3 Flash Semaphore Register

31	16
KEY	
R=0/W	

15	4	3	2	1	0
KEY		Reserved		SEM	
R=0/W		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-175. M3 Flash Semaphore Field Descriptions

Bit	Field	Value	Description
31-4	KEY	0	Writing the value 0xA5937EC will allow writes to the SEM bits or writes are ignored. Reads will return 0. **Note:** This is to prevent spurious writes to the semaphore bits.
3-2	Reserved		Reserved
1-0	SEM	0	Pump Access Request Semaphore from the M3 • A value of “00”, “10”, “11” gives ownership to the M3 • A value of “01” gives ownership to the C28. • The following are the only state transitions allowed on these bits. – "00","11" -> "01" – "00","11" -> "10" • "10" -> "00","11" (This can only happen from an M3 write to these bits to relinquish its ownership) • State transitions from "00" → "11" and "11" → "00" are allowed by design. However these transitions will not result in change in ownership.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: If simultaneous write access is performed to the above registers and a conflict occurs, the M3 will be given priority.

1.13.11.7 M3 Clock Semaphore Register

Figure 1-164. M3 Clock Semaphore Register

31															16
KEY															
R=0/W															
15												4	3	2	1	0
KEY													Reserved		SEM	
R=0/W													R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-176. M3 Flash Semaphore Field Descriptions

Bit	Field	Value	Description
31-4	KEY	0	Writing the value 0xB95C813 will allow writes to the SEM bits or writes are ignored. Reads will return 0. **Note:** This is to prevent spurious writes to the semaphore bits.
3-2	Reserved		Reserved
1-0	SEM	0	• A value of “00”, “10”, “11” gives ownership to the M3 • A value of “01” gives ownership to the C28. • The following are the only state transitions allowed on these bits. – "00","11" -> "01" – "00","11" -> "10" • "10" -> "00","11" (This can only happen from an M3 write to these bits to relinquish its ownership) • State transitions from "00" → "11" and "11" → "00" are allowed by design. However these transitions will not result in change in ownership.

1.13.12 Control Subsystem IPC Registers

The registers below are mapped to the control subsystem address map only.

1.13.12.1 CTOMIPCSET Register

Figure 1-165. CTOMIPCSET Register

31	30	29	28	27	26	25	24
IPC32	IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IPC16	IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-177. CTOMIPCSET Register Field Descriptions

Bit	Field	Value	Description
31	IPC32	0	CTOMIPCSET Flag 32. C28 to M3 core IPC flag 32 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
30	IPC31	0	CTOMIPCSET Flag 31. C28 to M3 core IPC flag 31 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
29	IPC30	0	CTOMIPCSET Flag 30. C28 to M3 core IPC flag 30 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers
28	IPC29	0	CTOMIPCSET Flag 29. C28 to M3 core IPC flag 29 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
27	IPC28	0	CTOMIPCSET Flag 28. C28 to M3 core IPC flag 28 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
26	IPC27	0	CTOMIPCSET Flag 27. C28 to M3 core IPC flag 27 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
25	IPC26	0	CTOMIPCSET Flag 26. C28 to M3 core IPC flag 26 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
24	IPC25	0	CTOMIPCSET Flag 25. C28 to M3 core IPC flag 25 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
23	IPC24	0	CTOMIPCSET Flag 24. C28 to M3 core IPC flag 24 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
22	IPC23	0	CTOMIPCSET Flag 23. C28 to M3 core IPC flag 23 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
21	IPC22	0	CTOMIPCSET Flag 22. C28 to M3 core IPC flag 22 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
20	IPC21	0	CTOMIPCSET Flag 21. C28 to M3 core IPC flag 21set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
19	IPC20	0	CTOMIPCSET Flag 20. C28 to M3 core IPC flag 20 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
18	IPC19	0	CTOMIPCSET Flag 19. C28 to M3 core IPC flag 19 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
17	IPC18	0	CTOMIPCSET Flag 18. C28 to M3 core IPC flag 18 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
16	IPC17	0	CTOMIPCSET Flag 17. C28 to M3 core IPC flag 17 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
15	IPC16	0	CTOMIPCSET Flag 16. C28 to M3 core IPC flag 16 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
14	IPC15	0	CTOMIPCSET Flag 15. C28 to M3 core IPC flag 15 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
13	IPC14	0	CTOMIPCSET Flag 14. C28 to M3 core IPC flag 14 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
12	IPC13	0	CTOMIPCSET Flag 13. C28 to M3 core IPC flag 13 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-177. CTOMIPCSET Register Field Descriptions (continued)

Bit	Field	Value	Description
11	IPC12	0	CTOMIPCSET Flag 12. C28 to M3 core IPC flag 12 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
10	IPC11	0	CTOMIPCSET Flag 11. C28 to M3 core IPC flag 11 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
9	IPC10	0	CTOMIPCSET Flag 10. C28 to M3 core IPC flag 10 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
8	IPC9	0	CTOMIPCSET Flag 9. C28 to M3 core IPC flag 9 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
7	IPC8	0	CTOMIPCSET Flag 8. C28 to M3 core IPC flag 8 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
6	IPC7	0	CTOMIPCSET Flag 7. C28 to M3 core IPC flag 7 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
5	IPC6	0	CTOMIPCSET Flag 6. C28 to M3 core IPC flag 6 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
4	IPC5	0	CTOMIPCSET Flag 5. C28 to M3 core IPC flag 5 set. If a bit is set by writing a '1' then the corresponding bit in CTOMIPCFLG is set. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
3	IPC4	0	CTOMIPCSET Interrupt 4. C28 to M3 IPC interrupt 4 request set. If this bit is set by writing a '1' then CTOMINT1 is raised to the M3 NVIC. The status of this bit is not readable in this register – it is readable in the corresponding bit in the CTOMIPCFLG and STS registers.
2	IPC3	0	CTOMIPCSET Interrupt 3. C28 to M3 IPC interrupt 3 request set. If this bit is set by writing a '1' then CTOMINT1 is raised to the M3 NVIC. The status of this bit is not readable in this register – it is readable in the corresponding bit in the CTOMIPCFLG and STS registers.
1	IPC2	0	CTOMIPCSET Interrupt 2. C28 to M3 IPC interrupt 2 request set. If this bit is set by writing a '1' then CTOMINT1 is raised to the M3 NVIC. The status of this bit is not readable in this register – it is readable in the corresponding bit in the CTOMIPCFLG and STS registers.
0	IPC1	0	CTOMIPCSET Interrupt 1. C28 to M3 IPC interrupt 1 request set. If this bit is set by writing a '1' then CTOMINT1 is raised to the M3 NVIC. The status of this bit is not readable in this register – it is readable in the corresponding bit in the CTOMIPCFLG and STS registers.

1.13.12.2 CTOMIPCCLR Register

Figure 1-166. CTOMIPCCLR Register

31	30	29	28	27	26	25	24
IPC32	IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IPC16	IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-178. CTOMIPCCLR Register Field Descriptions

Bit	Field	Value	Description
31	IPC32	0	CTOMIPCCLR Flag 32. C28 to M3 core IPC flag 32 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers
30	IPC31	0	CTOMIPCCLR Flag 31. C28 to M3 core IPC flag 31 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
29	IPC30	0	CTOMIPCCLR Flag 30. C28 to M3 core IPC flag 30 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
28	IPC29	0	CTOMIPCCLR Flag 29. C28 to M3 core IPC flag 29 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
27	IPC28	0	CTOMIPCCLR Flag 28. C28 to M3 core IPC flag 28 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
26	IPC27	0	CTOMIPCCLR Flag 27. C28 to M3 core IPC flag 27 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
25	IPC26	0	CTOMIPCCLR Flag 26. C28 to M3 core IPC flag 26 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
24	IPC25	0	CTOMIPCCLR Flag 25. C28 to M3 core IPC flag 25 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
23	IPC24	0	CTOMIPCCLR Flag 24. C28 to M3 core IPC flag 24 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
22	IPC23	0	CTOMIPCCLR Flag 23. C28 to M3 core IPC flag 23 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
21	IPC22	0	CTOMIPCCLR Flag 22. C28 to M3 core IPC flag 22 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
20	IPC21	0	CTOMIPCCLR Flag 21. C28 to M3 core IPC flag 21clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
19	IPC20	0	CTOMIPCCLR Flag 20. C28 to M3 core IPC flag 20 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
18	IPC19	0	CTOMIPCCLR Flag 19. C28 to M3 core IPC flag 19 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
17	IPC18	0	CTOMIPCCLR Flag 18. C28 to M3 core IPC flag 18 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
16	IPC17	0	CTOMIPCCLR Flag 17. C28 to M3 core IPC flag 17 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
15	IPC16	0	CTOMIPCCLR Flag 16. C28 to M3 core IPC flag 16 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers
14	IPC15	0	CTOMIPCCLR Flag 15. C28 to M3 core IPC flag 15 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
13	IPC14	0	CTOMIPCCLR Flag 14. C28 to M3 core IPC flag 14 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
12	IPC13	0	CTOMIPCCLR Flag 13. C28 to M3 core IPC flag 13 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-178. CTOMIPCCLR Register Field Descriptions (continued)

Bit	Field	Value	Description
11	IPC12	0	CTOMIPCCLR Flag 12. C28 to M3 core IPC flag 12 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
10	IPC11	0	CTOMIPCCLR Flag 11. C28 to M3 core IPC flag 11 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
9	IPC10	0	CTOMIPCCLR Flag 10. C28 to M3 core IPC flag 10 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
8	IPC9	0	CTOMIPCCLR Flag 9. C28 to M3 core IPC flag 9 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
7	IPC8	0	CTOMIPCCLR Flag 8. C28 to M3 core IPC flag 8 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
6	IPC7	0	CTOMIPCCLR Flag 7. C28 to M3 core IPC flag 7 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
5	IPC6	0	CTOMIPCCLR Flag 6. C28 to M3 core IPC flag 6 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
4	IPC5	0	CTOMIPCCLR Flag 5. C28 to M3 core IPC flag 5 clear. If a bit is cleared by writing a '1' then the corresponding bit in CTOMIPCFLG is cleared. The status of this bit is not readable in this register – it is readable in the CTOMIPCFLG and STS registers.
3	IPC4	0	CTOMIPCCLR Interrupt 4. C28 to M3 IPC interrupt 4 request clear. If this bit is cleared by writing a '1' then CTOMINT1 is raised to the M3 NVIC. The status of this bit is not readable in this register – it is readable in the corresponding bit in the CTOMIPCFLG and STS registers.
2	IPC3	0	CTOMIPCCLR Interrupt 3. C28 to M3 IPC interrupt 3 request clear. If this bit is cleared by writing a '1' then CTOMINT1 is raised to the M3 NVIC. The status of this bit is not readable in this register – it is readable in the corresponding bit in the CTOMIPCFLG and STS registers.
1	IPC2	0	CTOMIPCCLR Interrupt 2. C28 to M3 IPC interrupt 2 request clear. If this bit is cleared by writing a '1' then CTOMINT1 is raised to the M3 NVIC. The status of this bit is not readable in this register – it is readable in the corresponding bit in the CTOMIPCFLG and STS registers.
0	IPC1	0	CTOMIPCCLR Interrupt 1. C28 to M3 IPC interrupt 1 request clear. If this bit is cleared by writing a '1' then CTOMINT1 is raised to the M3 NVIC. The status of this bit is not readable in this register – it is readable in the corresponding bit in the CTOMIPCFLG and STS registers.

1.13.12.3 CTOMIPCFLG Register

Figure 1-167. CTOMIPCFLG Register

31	30	29	28	27	26	25	24
IPC32	IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17
RR-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
IPC16	IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-179. CTOMIPCFLG Register Field Descriptions

Bit	Field	Value	Description
31	IPC32	0	CTOMIPCFLG Flag 32. C28 to M3 core IPC flag 32 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
30	IPC31	0	CTOMIPCFLG Flag 31. C28 to M3 core IPC flag 31 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
29	IPC30	0	CTOMIPCFLG Flag 30. C28 to M3 core IPC flag 30 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
28	IPC29	0	CTOMIPCFLG Flag 29. C28 to M3 core IPC flag 29 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
27	IPC28	0	CTOMIPCFLG Flag 28. C28 to M3 core IPC flag 28 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
26	IPC27	0	CTOMIPCFLG Flag 27. C28 to M3 core IPC flag 27 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
25	IPC26	0	CTOMIPCFLG Flag 26. C28 to M3 core IPC flag 26 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
24	IPC25	0	CTOMIPCFLG Flag 25. C28 to M3 core IPC flag 25 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
23	IPC24	0	CTOMIPCFLG Flag 24. C28 to M3 core IPC flag 24 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
22	IPC23	0	CTOMIPCFLG Flag 23. C28 to M3 core IPC flag 23 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
21	IPC22	0	CTOMIPCFLG Flag 22. C28 to M3 core IPC flag 22 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
20	IPC21	0	CTOMIPCFLG Flag 21. C28 to M3 core IPC flag 21 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
19	IPC20	0	CTOMIPCFLG Flag 20. C28 to M3 core IPC flag 20 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
18	IPC19	0	CTOMIPCFLG Flag 19. C28 to M3 core IPC flag 19 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
17	IPC18	0	CTOMIPCFLG Flag 18. C28 to M3 core IPC flag 18 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
16	IPC17	0	CTOMIPCFLG Flag 17. C28 to M3 core IPC flag 17 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
15	IPC16	0	CTOMIPCFLG Flag 16. C28 to M3 core IPC flag 16 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
14	IPC15	0	CTOMIPCFLG Flag 15. C28 to M3 core IPC flag 15 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
13	IPC14	0	CTOMIPCFLG Flag 14. C28 to M3 core IPC flag 14 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
12	IPC13	0	CTOMIPCFLG Flag 13. C28 to M3 core IPC flag 13 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-179. CTOMIPCFLG Register Field Descriptions (continued)

Bit	Field	Value	Description
11	IPC12	0	CTOMIPCFLG Flag 12. C28 to M3 core IPC flag 12 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
10	IPC11	0	CTOMIPCFLG Flag 11. C28 to M3 core IPC flag 11 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
9	IPC10	0	CTOMIPCFLG Flag 10. C28 to M3 core IPC flag 10 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
8	IPC9	0	CTOMIPCFLG Flag 9. C28 to M3 core IPC flag 9 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
7	IPC8	0	CTOMIPCFLG Flag 8. C28 to M3 core IPC flag 8 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
6	IPC7	0	CTOMIPCFLG Flag 7. C28 to M3 core IPC flag 7 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
5	IPC6	0	CTOMIPCFLG Flag 6. C28 to M3 core IPC flag 6 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
4	IPC5	0	CTOMIPCFLG Flag 5. C28 to M3 core IPC flag 5 status. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’.
3	IPC4	0	CTOMIPCFLG Interrupt 4. C28 to M3 IPC interrupt 4 status flag. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’
2	IPC3	0	CTOMIPCFLG Interrupt 3. C28 to M3 IPC interrupt 3 status flag. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’
1	IPC2	0	CTOMIPCFLG Interrupt 2. C28 to M3 IPC interrupt 2 status flag. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’
0	IPC1	0	CTOMIPCFLG Interrupt 1. C28 to M3 IPC interrupt 1 status flag. The bit is ‘1’ if the corresponding CTOMIPCSET bit has been written with a ‘1’ and CTOMIPCCLR or CTOMIPCACK bit has not been written with a ‘1’

1.13.12.4 MTOCIPCACK Register

Figure 1-168. MTOCIPCACK Register

31	30	29	28	27	26	25	24
IPC32	IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IPC16	IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-180. MTOCIPCACK Register Field Descriptions

Bit	Field	Value	Description
31	IPC32	0	MTOCIPCACK Flag 32. M3 to C28 core IPC flag 32 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
30	IPC31	0	MTOCIPCACK Flag 31. M3 to C28 core IPC flag 31 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
29	IPC30	0	MTOCIPCACK Flag 30. M3 to C28 core IPC flag 30 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
28	IPC29	0	MTOCIPCACK Flag 29. M3 to C28 core IPC flag 29 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
27	IPC28	0	MTOCIPCACK Flag 28. M3 to C28 core IPC flag 28 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
26	IPC27	0	MTOCIPCACK Flag 27. M3 to C28 core IPC flag 27 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
25	IPC26	0	MTOCIPCACK Flag 26. M3 to C28 core IPC flag 26 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
24	IPC25	0	MTOCIPCACK Flag 25. M3 to C28 core IPC flag 25 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
23	IPC24	0	MTOCIPCACK Flag 24. M3 to C28 core IPC flag 24 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
22	IPC23	0	MTOCIPCACK Flag 23. M3 to C28 core IPC flag 23 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
21	IPC22	0	MTOCIPCACK Flag 22. M3 to C28 core IPC flag 22 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
20	IPC21	0	MTOCIPCACK Flag 21. M3 to C28 core IPC flag 21 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
19	IPC20	0	MTOCIPCACK Flag 20. M3 to C28 core IPC flag 20 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
18	IPC19	0	MTOCIPCACK Flag 19. M3 to C28 core IPC flag 19 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
17	IPC18	0	MTOCIPCACK Flag 18. M3 to C28 core IPC flag 18 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
16	IPC17	0	MTOCIPCACK Flag 17. M3 to C28 core IPC flag 17 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
15	IPC16	0	MTOCIPCACK Flag 16. M3 to C28 core IPC flag 16 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
14	IPC15	0	MTOCIPCACK Flag 15. M3 to C28 core IPC flag 15 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
13	IPC14	0	MTOCIPCACK Flag 14. M3 to C28 core IPC flag 14 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
12	IPC13	0	MTOCIPCACK Flag 13. M3 to C28 core IPC flag 13 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-180. MTOCIPCACK Register Field Descriptions (continued)

Bit	Field	Value	Description
11	IPC12	0	MTOCIPCACK Flag 12. M3 to C28 core IPC flag 12 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
10	IPC11	0	MTOCIPCACK Flag 11. M3 to C28 core IPC flag 11 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
9	IPC10	0	MTOCIPCACK Flag 10. M3 to C28 core IPC flag 10 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
8	IPC9	0	MTOCIPCACK Flag 9. M3 to C28 core IPC flag 9 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
7	IPC8	0	MTOCIPCACK Flag 8. M3 to C28 core IPC flag 8 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
6	IPC7	0	MTOCIPCACK Flag 7. M3 to C28 core IPC flag 7 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
5	IPC6	0	MTOCIPCACK Flag 6. M3 to C28 core IPC flag 6 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
4	IPC5	0	MTOCIPCACK Flag 5. M3 to C28 core IPC flag 5 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
3	IPC4	0	MTOCIPCACK Interrupt 4. M3 to C28 IPC interrupt 4 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
2	IPC3	0	MTOCIPCACK Interrupt 3. M3 to C28 IPC interrupt 3 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
1	IPC2	0	MTOCIPCACK Interrupt 2. M3 to C28 IPC interrupt 2 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.
0	IPC1	0	MTOCIPCACK Interrupt 1. M3 to C28 IPC interrupt 1 acknowledge. Writing a '1' to this bit clears the corresponding bit in MTOCIPCFLG and MTOCIPCSTS to '0'. The status of this bit is not readable in this register – it is readable in the MTOCIPCFLG and STS registers.

1.13.12.5 MTOCIPCSTS Register

Figure 1-169. MTOCIPCSTS Register

31	30	29	28	27	26	25	24
IPC32	IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
IPC16	IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-181. MTOCIPCSTS Register Field Descriptions

Bit	Field	Value	Description
31	IPC32	0	MTOCIPCSTS Flag 32. M3 to C28 core IPC flag 32 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
30	IPC31	0	MTOCIPCSTS Flag 31. M3 to C28 core IPC flag 31 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
29	IPC30	0	MTOCIPCSTS Flag 30. M3 to C28 core IPC flag 30 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
28	IPC29	0	MTOCIPCSTS Flag 29. M3 to C28 core IPC flag 29 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
27	IPC28	0	MTOCIPCSTS Flag 28. M3 to C28 core IPC flag 28 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
26	IPC27	0	MTOCIPCSTS Flag 27. M3 to C28 core IPC flag 27 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
25	IPC26	0	MTOCIPCSTS Flag 26. M3 to C28 core IPC flag 26 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
24	IPC25	0	MTOCIPCSTS Flag 25. M3 to C28 core IPC flag 25 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
23	IPC24	0	MTOCIPCSTS Flag 24. M3 to C28 core IPC flag 24 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
22	IPC23	0	MTOCIPCSTS Flag 23. M3 to C28 core IPC flag 23 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
21	IPC22	0	MTOCIPCSTS Flag 22. M3 to C28 core IPC flag 22 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
20	IPC21	0	MTOCIPCSTS Flag 21. M3 to C28 core IPC flag 21 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
19	IPC20	0	MTOCIPCSTS Flag 20. M3 to C28 core IPC flag 20 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
18	IPC19	0	MTOCIPCSTS Flag 19. M3 to C28 core IPC flag 19 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
17	IPC18	0	MTOCIPCSTS Flag 18. M3 to C28 core IPC flag 18 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
16	IPC17	0	MTOCIPCSTS Flag 17. M3 to C28 core IPC flag 17 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
15	IPC16	0	MTOCIPCSTS Flag 16. M3 to C28 core IPC flag 16 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
14	IPC15	0	MTOCIPCSTS Flag 15. M3 to C28 core IPC flag 15 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
13	IPC14	0	MTOCIPCSTS Flag 14. M3 to C28 core IPC flag 14 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
12	IPC13	0	MTOCIPCSTS Flag 13. M3 to C28 core IPC flag 13 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.

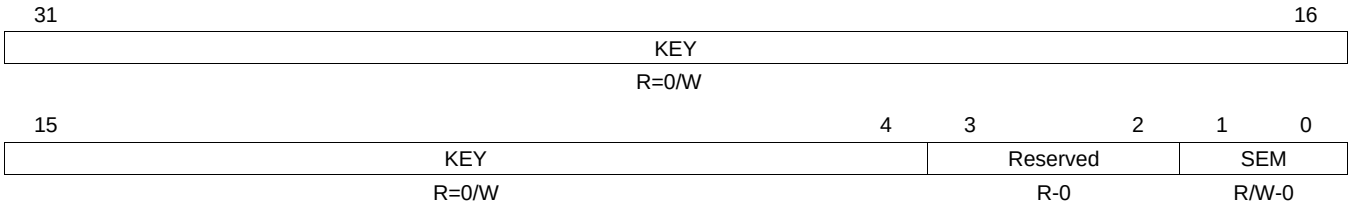
Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-181. MTOCIPCSTS Register Field Descriptions (continued)

Bit	Field	Value	Description
11	IPC12	0	MTOCIPCSTS Flag 12. M3 to C28 core IPC flag 12 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
10	IPC11	0	MTOCIPCSTS Flag 11. M3 to C28 core IPC flag 11 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
9	IPC10	0	MTOCIPCSTS Flag 10. M3 to C28 core IPC flag 10 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
8	IPC9	0	MTOCIPCSTS Flag 9. M3 to C28 core IPC flag 9 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
7	IPC8	0	MTOCIPCSTS Flag 8. M3 to C28 core IPC flag 8 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
6	IPC7	0	MTOCIPCSTS Flag 7. M3 to C28 core IPC flag 7 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
5	IPC6	0	MTOCIPCSTS Flag 6. M3 to C28 core IPC flag 6 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
4	IPC5	0	MTOCIPCSTS Flag 5. M3 to C28 core IPC flag 5 status. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’.
3	IPC4	0	MTOCIPCSTS Interrupt 4. M3 to C28 IPC interrupt 4 status flag. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’
2	IPC3	0	MTOCIPCSTS Interrupt 3. M3 to C28 IPC interrupt 3 status flag. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1’
1	IPC2	0	MTOCIPCSTS Interrupt 2. M3 to C28 IPC interrupt 2 status flag. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1.’
0	IPC1	0	MTOCIPCSTS Interrupt 1. M3 to C28 IPC interrupt 1 status flag. The bit is ‘1’ if the corresponding MTOCIPCSET bit has been written with a ‘1’ and MTOCIPCCLR or MTOCIPCACK bit has not been written with a ‘1.’

1.13.12.6 C28 Flash Semaphore Register

Figure 1-170. C28 Flash Semaphore Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-182. C28 Flash Semaphore Field Descriptions

Bit	Field	Value	Description
31-4	KEY	0	Writing the value 0x4CE7395 will allow writes to the SEM bits or writes are ignored. Reads will return 0. **Note:** This is to prevent spurious writes to the semaphore bits.
3-2	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-182. C28 Flash Semaphore Field Descriptions (continued)

Bit	Field	Value	Description
1-0	SEM	0	Pump Access Request Semaphore from the C28 • A value of “00”, “10”, “11” gives ownership to the M3 • A value of “01” gives ownership to the C28. • The following are the only state transitions allowed on these bits. – "00","11" -> "01" – "00","11" -> "10" • "10" -> "00","11" (This can only happen from a C28x write to these bits to relinquish its ownership) • State transitions from "00" → "11" and "11" → "00" are allowed by design. However these transitions will not result in change in ownership.

1.13.12.7 C28 Clock Semaphore Register

Figure 1-171. C28 Clock Semaphore Register

31	16
KEY	
R=0/W	

15	4	3	2	1	0
KEY		Reserved		SEM	
R=0/W		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-183. C28 Clock Semaphore Field Descriptions

Bit	Field	Value	Description
31-4	KEY	0	Writing the value 0xE318C59 will allow writes to the SEM bits or writes are ignored. Reads will return 0. **Note:** This is to prevent spurious writes to the semaphore bits.
3-2	Reserved		Reserved
1-0	SEM	0	Clock Control Access Request from the C28 • A value of “00”, “10”, “11” gives ownership to the M3 • A value of “01” gives ownership to the C28. • The following are the only state transitions allowed on these bits. – "00","11" -> "01" – "00","11" -> "10" • "10" -> "00","11" (This can only happen from a C28x write to these bits to relinquish its ownership) • State transitions from "00" → "11" and "11" → "00" are allowed by design. However these transitions will not result in change in ownership.

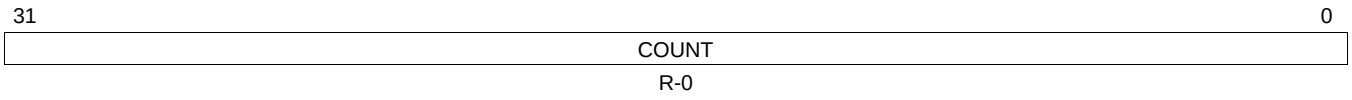
Copyright © 2012–2019, Texas Instruments Incorporated

1.13.13 Master and Control Subsystem IPC Registers

The below registers are dual-mapped to both the master and control subsystems. For read/write registers, see the register description to determine which subsystem has read/write or read-only access to the register.

1.13.13.1 MIPCCOUNTERL Register

Figure 1-172. MIPCCOUNTERL Register



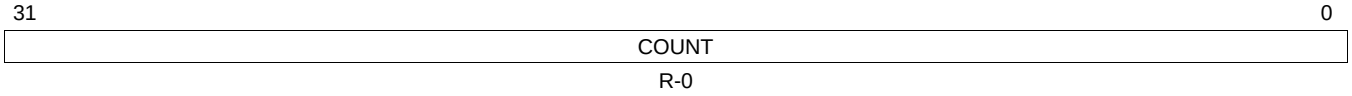
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-184. MIPCCOUNTERL Register Field Descriptions

Bit	Field	Value	Description
31-0	COUNT	0	C28 IPC Counter Register. This is the low 32-bits of the free running 64 bit timestamp counter clocked by the shared resource clock.

1.13.13.2 MIPCCOUNTERH Register

Figure 1-173. MIPCCOUNTERH Register



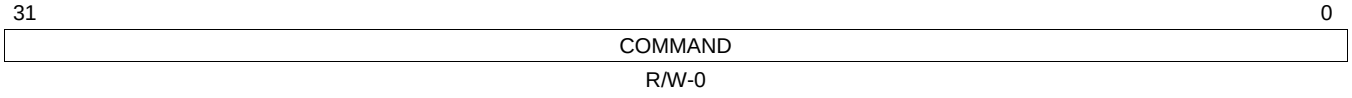
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-185. MIPCCOUNTERH Register Field Descriptions

Bit	Field	Value	Description
31-0	COUNT	0	M3 IPC Counter Register. This is the high 32-bits of the free running 64 bit timestamp counter clocked by the shared resource clock.

1.13.13.3 CTOMIPCCOM Register

Figure 1-174. CTOMIPCCOM Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-186. CTOMIPCCOM Register Field Descriptions

Bit	Field	Value	Description
31-0	COMMAND	0	C28 TO M3 IPC Command Register. This register is defined and interpreted by software – used as command place holder for IPC commands from the C28 to the M3 CPU. It is read/write for the C28 CPU and read only for the M3 CPU.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.13.4 CTOMIPCADDR Register

Figure 1-175. CTOMIPCADDR Register

31	0
ADDR	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-187. CTOMIPCADDR Register Field Descriptions

Bit	Field	Value	Description
31-0	ADDR	0	C28 TO M3 IPC Address Register. This register is used as an address holder for IPC commands from the C28 to the M3 CPU. It is read/write to the C28 CPU and read only to the M3 CPU.

1.13.13.5 CTOMIPCDATAW Register

Figure 1-176. CTOMIPCDATAW Register

31	0
WDATA	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-188. CTOMIPCDATAW Register Field Descriptions

Bit	Field	Value	Description
31-0	WDATA	0	C28 TO M3 IPC Data Write Registe. This register is used as a write data holder for IPC commands from the C28 to the M3 CPU. It is read/write to the C28 CPU and read only to the M3 CPU.

1.13.13.6 CTOMIPCDATAR Register

Figure 1-177. CTOMIPCDATAR Register

31	0
RDATA	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-189. CTOMIPCDATAR Register Field Descriptions

Bit	Field	Value	Description
31-0	RDATA	0	C28 TO M3 IPC Data Read Register. This register is used as a read data holder for IPC commands from the C28 to the M3 CPU. It is read/write to the M3 CPU and read only to the C28 CPU.

1.13.13.7 MTOCIPCCOM Register

Figure 1-178. MTOCIPCCOM Register

31	0
COMMAND	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-190. MTOCIPCCOM Register Field Descriptions

Bit	Field	Value	Description
31-0	COMMAND	0	M3 TO C28 IPC Command Register. This register is defined and interpreted by software – used as command place holder for IPC commands from the M3 to the C28 CPU. It is read/write to the M3 CPU and read only to the C28 CPU.

Copyright © 2012–2019, Texas Instruments Incorporated

1.13.13.8 MTOCIPCADDR Register

Figure 1-179. MTOCIPCADDR Register

31	0
ADDR	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-191. MTOCIPCADDR Register Field Descriptions

Bit	Field	Value	Description
31-0	ADDR	0	M3 TO C28 IPC Address Register. This register used as an address holder for IPC commands from the M3 to the C28 CPU. It is read/write to the M3 CPU and read only to the C28 CPU.

1.13.13.9 MTOCIPCDATAW Register

Figure 1-180. MTOCIPCDATAW Register

31	0
WDATA	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-192. MTOCIPCDATAW Register Field Descriptions

Bit	Field	Value	Description
31-0	WDATA	0	M3 TO C28 IPC Data Write Register. This register is used as a write data holder for IPC commands from the M3 to the C28 CPU. It is read/write to the M3 CPU and read only to the C28 CPU.

1.13.13.10 MTOCIPCDATAR Register

Figure 1-181. MTOCIPCDATAR Register

31	0
RDATA	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-193. MTOCIPCDATAR Register Field Descriptions

Bit	Field	Value	Description
31-0	RDATA	0	M3 TO C28 IPC Data Read Register. This register is an IPC read data hold register for the M3 to the C28 CPU IPC. It is read/write to the C28 CPU and read only to the M3 CPU.

1.13.13.11 CTOMIPCBOOTSTS Register

Figure 1-182. CTOMIPCBOOTSTS Register

31	0
BOOTSTS	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 1-194. CTOMIPCBOOTSTS Register Field Descriptions

Bit	Field	Value	Description
31-0	BOOTSTS	0	C28 TO M3 IPC Boot Status Register. This is an IPC boot status register where C28 boot ROM writes the boot status code that can be read by the M3 to report the boot condition of the C28. It is read/write to the C28 CPU and read only to the M3 CPU

1.13.13.12 MTOCIPCBOOTMODE Register

Figure 1-183. MTOCIPCBOOTMODERegister

31	0
BOOTMODE	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 1-195. MTOCIPCBOOTMODE Register Field Descriptions

Bit	Field	Value	Description
31-0	BOOTMODE	0	M3 TO C28 IPC Boot Mode Register. This is an IPC boot mode register where the M3 can specify the C28 boot mode to enter. It is read/write to the M3 CPU and read only to the C28 CPU.

Copyright © 2012–2019, Texas Instruments Incorporated

M3 General-Purpose Timers

This chapter discusses the general-purpose timer module (GPTM). Programmable timers can be used to count or to time external events that drive the timer input pins. This module contains four GPTM blocks. Each GPTM block provides two 16-bit timers/counters (referred to as Timer A and Timer B) that can be configured to operate independently as timers or event counters, or concatenated to operate as one 32-bit timer or one 32-bit real-time clock (RTC). Timers can also be used to trigger µDMA transfers.

The GPT Module is one timing resource available on the microcontrollers; another is the system timer (SysTick) (see the Cortex-M3 Peripherals chapter) .

Topic **Page**

2.1 GPTM Features **299**
2.2 Block Diagram **299**
2.3 Functional Description **300**
2.4 Initialization and Configuration **306**
2.5 Register Map **308**
2.6 Register Descriptions **309**

Copyright © 2012–2019, Texas Instruments Incorporated

2.1 GPTM Features

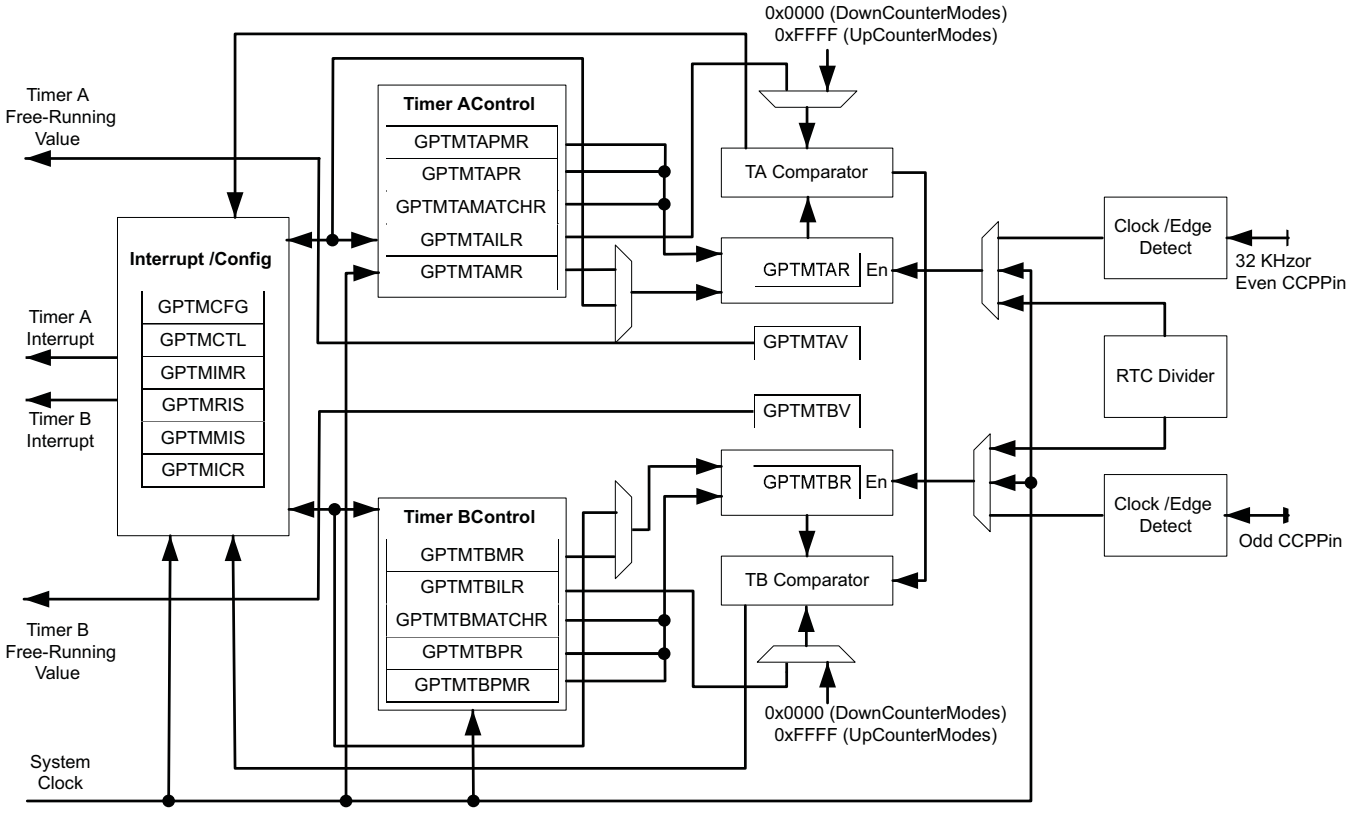
This module contains four GPTM blocks with the following functional options:

- Operating modes:
  - 16- or 32-bit programmable one-shot timer
  - 16- or 32-bit programmable periodic timer
  - 16-bit general-purpose timer with an 8-bit prescaler
  - 32-bit real-time clock (RTC) when using an external 32.768-KHz clock as the input
  - 16-bit input-edge count- or time-capture modes
  - 16-bit PWM mode with software-programmable output inversion of the PWM signal
- Count up or down
- Eight capture compare PWM pins (CCP)
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- 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

2.2 Block Diagram

In the block diagram (Figure 2-1), the specific capture compare PWM (CCP) pins available depend on the device. See Table 2-1 for the available CCP pins and their timer assignments.

Figure 2-1. GPTM Block Diagram



Copyright © 2012–2019, Texas Instruments Incorporated

Table 2-1. Available CCP Pins

Timer	16-Bit Up/Down Counter	Even CCP Pin	Odd CCP Pin
Timer 0	TimerA	CCP0	-
	TimerB	-	CCP1
Timer 1	TimerA	CCP2	-
	TimerB	-	CCP3
Timer 2	TimerA	CCP4	-
	TimerB	-	CCP5
Timer 3	TimerA	CCP6	-
	TimerB	-	CCP7

2.3 Functional Description

The main components of each GPTM block are two free-running up/down counters (referred to as Timer A and Timer B), two match registers, two prescaler match registers, two shadow registers, and two load/initialization registers and their associated control functions. The exact functionality of each GPTM is controlled by software and configured through the register interface. Timer A and Timer B can be used individually, in which case they have a 16-bit counting range. In addition, Timer A and Timer B can be concatenated to provide a 32-bit counting range. Note that the prescaler can only be used when the timers are used individually.

The available modes for each GPTM block are shown in Table 2-2. Note that when counting down, the prescaler acts as a true prescaler and contains the least-significant bits of the count. When counting up, the prescaler acts as a timer extension and holds the most-significant bits of the count

Table 2-2. General-Purpose Timer Capabilities

Mode	Timer Use	Count Direction	Counter Size	Prescaler Size[(1)]
One-shot	Individual	Up or Down	16-bit	8-bit
	Concatenated	Up or Down	32-bit	-
Periodic	Individual	Up or Down	16-bit	8-bit
	Concatenated	Up or Down	32-bit	-
RTC	Concatenated	Up	32-bit	-
Edge Count	Individual	Down	16-bit	8-bit
Edge Time	Individual	Down	16-bit	-
PWM	Individual	Down	16-bit	-

(1) The prescaler is only available when the timers are used individually

Software configures the GPTM using the GPTM Configuration (GPTMCFG) register, the GPTM Timer A Mode (GPTMTAMR) register, and the GPTM Timer B Mode (GPTMTBMR) register. When in one of the concatentated modes, Timer A and Timer B can only operate in one mode. However, when configured in an individual mode, Timer A and Timer B can be independently configured in any combination of the individual modes.

2.3.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters Timer A and Timer B are initialized to all 1s, along with their corresponding load registers: the GPTM Timer A Interval Load (GPTMTAILR) register and the GPTM Timer B Interval Load (GPTMTBILR) register; and shadow registers: the GPTM Timer A Value (GPTMTAV) register and the GPTM Timer B Value (GPTMTBV) register. The prescale counters are initialized to 0x00: the GPTM Timer A Prescale (GPTMTAPR) register, and the GPTM Timer B Prescale (GPTMTBPR) register .

Copyright © 2012–2019, Texas Instruments Incorporated

2.3.2 Timer Modes

This section describes the operation of the various timer modes. When using Timer A and Timer B in concatenated mode, only the Timer A control and status bits must be used; there is no need to use Timer B control and status bits. The GPTM is placed into individual mode by writing a value of 0x4 to the GPTM Configuration (GPTMCFG) register. In the following sections, the variable "n" is used in bit field and register names to imply either a Timer A function or a Timer B function. The prescaler is only available in the 16-bit one-shot, periodic, and input edge count timer mode. Throughout this section, the timeout event in down-count mode is 0x0 and in up-count mode is the value in the GPTM Timer n Match (GPTMTnMATCH) and the optional GPTM Timer n Prescale Match (GPTMTnPMR) registers.

2.3.2.1 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. 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, 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.

When the timer is counting down and it reaches the timeout event (0x0), the timer reloads its start value from the GPTMTnILR and the GPTMTnPR registers on the next cycle. When the timer is counting up and it reaches the timeout event (the value in the GPTMTnILR and the GPTMTnPR registers), the timer reloads with 0x0. If configured to be a one-shot timer, the timer stops counting and clears the TnEN bit in the GPTMCTL register. If configured as a periodic timer, the timer starts counting again on the next cycle. In periodic, snap-shot mode (TnSNAPS bit in the GPTMTnMR register is set), the actual free-running value of the timer at the time-out event is loaded into the GPTMTnR register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry.

In addition to reloading the count value, the GPTM generates interrupts and triggers when it reaches the time-out event. The GPTM sets the TnTORIS bit in the GPTM Raw Interrupt Status (GPTMRIS) register and holds it until it is cleared by writing the GPTM Interrupt Clear (GPTMICR) register. If the timeout interrupt is enabled in the GPTM Interrupt Mask (GPTMIMR) register, the GPTM also sets the TnTOMIS bit in the GPTM Masked Interrupt Status (GPTMMIS) register. By setting the TnMIE bit in the GPTMTAMR register, an interrupt can also be generated when the Timer value equals the value loaded into the GPTM Timer n Match (GPTMTnMATCH) and GPTM Timer n Prescale Match (GPTMTnPMR) registers. This interrupt has the same status, masking, and clearing functions as the timeout interrupt. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See the *Micro Direct Memory Access (µDMA)* chapter.

If software updates the GPTMTnILR register while the counter is counting down, the counter loads the new value on the next clock cycle and continues counting down from the new value. If software updates the GPTMTnILR register while the counter is counting up, the timeout event is changed on the next cycle to the new value. If software updates the GPTM Timer n Value (GPTMTnV) register while the counter is counting up or down, the counter loads the new value on the next clock cycle and continues counting from the new value. If software updates the GPTMTnMATCHR register while the counter is counting, the counter loads the new value on the next clock cycle and continues counting from the new value.

If the TnSTALL bit in the GPTMCTL register is set, the timer freezes counting while the processor is halted by the debugger. The timer resumes counting when the processor resumes execution.

The following table shows a variety of configurations for a 16-bit free-running timer while using the prescaler. All values assume a 100-MHz clock with Tc=10 ns (clock period).

Table 2-3. 16-Bit Timer With Prescaler Configurations

Prescale	#Clock (Tc)	Max Time	Units
00000000	1	0.6554	mS
00000001	2	1.3107	mS
00000010	3	1.9661	mS
----------	--	--	--

Copyright © 2012–2019, Texas Instruments Incorporated

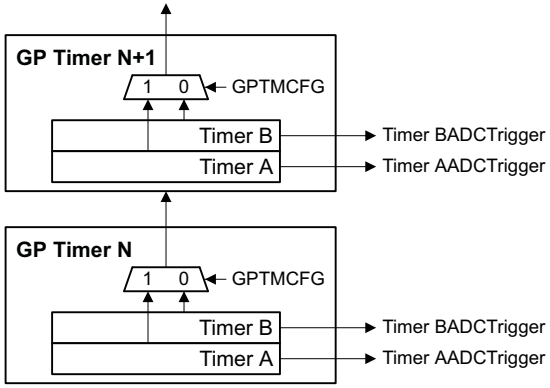
Table 2-3. 16-Bit Timer With Prescaler Configurations (continued)

Prescale	#Clock (Tc)	Max Time	Units
11111101	254	166.4614	mS
11111110	255	167.1168	mS
11111111	256	167.7722	mS

2.3.2.1.1 Wait-for-Trigger Mode

The wait-for-trigger mode allows daisy chaining of the timer modules such that once configured, a single timer can initiate mulitple timing events using the Timer triggers. Wait-for-trigger mode is enabled by setting the TnWOT bit in the GPTMTnMR register. When the TnWOT bit is set, Timer N+1 does not begin counting until the timer in the previous position in the daisy chain (Timer N) reaches its time-out event. The daisy chain is configured such that GPTM1 always follows GPTM0, GPTM2 follows GPTM1, and so on. If Timer A is in 32-bit mode (controlled by the GPTMCFG bit in the GPTMCFG register), it triggers Timer A in the next module. If Timer A is in 16-bit mode, it triggers Timer B in the same module, and Timer B triggers Timer A in the next module. Care must be taken that the TAWOT bit is never set in GPTM0. Figure 2-2 shows how the GPTMCFG bit affects the daisy chain. This function is valid for both one-shot and periodic modes.

Figure 2-2. Timer Daisy Chain



2.3.2.2 Real-Time Clock Timer Mode

In real-time clock (RTC) mode, the concatenated versions of the Timer A and Timer B registers are configured as an up-counter. When RTC mode is selected for the first time after reset, the counter is loaded with a value of 0x1. All subsequent load values must be written to the GPTM Timer A Interval Load (GPTMTAILR) register.

The input clock on an even CCP input is required to be 32.768 KHz in RTC mode. The clock signal is then divided down to a 1-Hz rate and is passed along to the input of the counter.

When software writes the TAEN bit in the GPTMCTL register, the counter starts counting up from its preloaded value of 0x1. When the current count value matches the preloaded value in the GPTMTAMATCHR register, the GPTM asserts the RTCRIS bit in GPTMRIS and continues counting until either a hardware reset, or it is disabled by software (clearing the TAEN bit). When the timer value reaches the terminal count, the timer rolls over and continues counting up from 0x0. If the RTC interrupt is enabled in GPTMIMR, the GPTM also sets the RTCMIS bit in GPTMMIS and generates a controller interrupt. The status flags are cleared by writing the RTCCINT bit in GPTMICR.

In addition to generating interrupts, a µMA trigger can be generated. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See Channel Configuration in the *Micro Direct Memory Access (µDMA)* chapter.

If the TASTALL and/or TBSTALL bits in the GPTMCTL register are set, the timer does not freeze if the RTCEN bit is set in GPTMCTL.

Copyright © 2012–2019, Texas Instruments Incorporated

2.3.2.3 Input Edge-Count Mode

NOTE: For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

In edge-count mode, the timer is configured as a 24-bit down-counter including the optional prescaler with the upper count value stored in the GPTM Timer n Prescale (GPTMTnPR) register and the lower bits in the GPTMTnILR register. In this mode, the timer is capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge-Count mode, the TnCMR bit of the GPTMTnMR register must be cleared. The type of edge that the timer counts is determined by the TnEVENT fields of the GPTMCTL register. During initialization, the GPTMTnMATCHR and GPTMTnPMR registers are configured so that the difference between the value in the GPTMTnILR and GPTMTnPR registers and the GPTMTnMATCHR and GPTMTnPMR registers equals the number of edge events that must be counted.

When software writes the TnEN bit in the GPTM Control (GPTMCTL) register, the timer is enabled for event capture. Each input event on the CCP pin decrements the counter by 1 until the event count matches GPTMTnMATCHR and GPTMTnPMR. When the counts match, the GPTM asserts the CnMRIS bit in the GPTMRIS register (and the CnMMIS bit, if the interrupt is not masked).

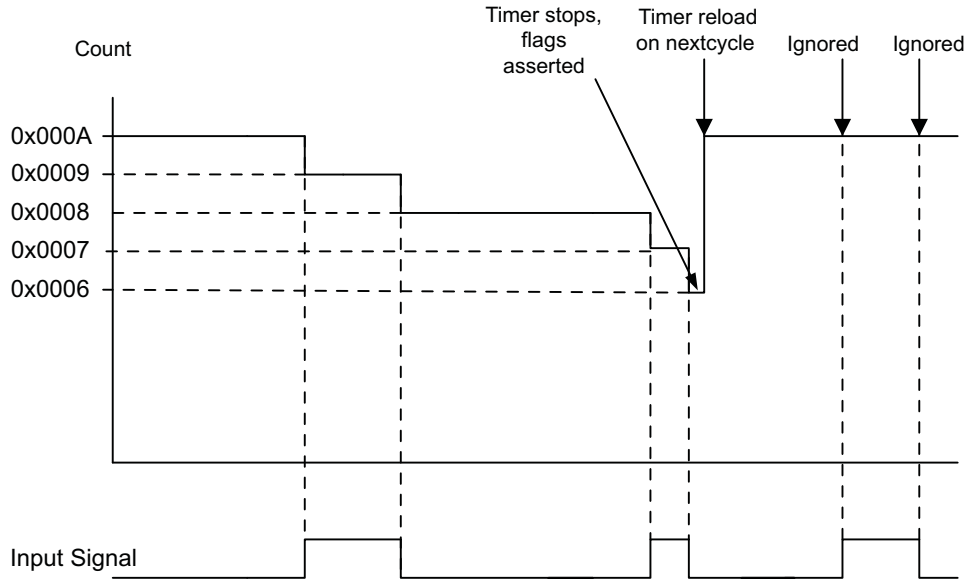
In addition to generating interrupts, a µDMA trigger can be generated. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See Channel Configuration in the *Micro Direct Memory Access (µDMA)* chapter.

After the match value is reached, the counter is then reloaded using the value in GPTMTnILR and GPTMTnPR registers, and stopped because the GPTM automatically clears the TnEN bit in the GPTMCTL register. Once the event count has been reached, all further events are ignored until TnEN is re-enabled by software.

Figure 2-3 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 2-3. Edge-Count Mode Example



Copyright © 2012–2019, Texas Instruments Incorporated

2.3.2.4 Input Edge-Time Mode

NOTE: For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling edge detection, the input signal must be low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

The prescaler is not available in 16-Bit Input edge-time mode.

In edge-time mode, the timer is configured as a 16-bit down-counter. In this mode, the timer is initialized to the value loaded in the GPTMTnILRregister. The timer is capable of capturing three types of events: rising edge, falling edge, or both. The timer is placed into edge-time mode by setting the TnCMR bit in the GPTMTnMR register, and the type of event that the timer captures is determined by the TnEVENT fields of the GPTMCTL register.

When software writes the TnEN bit in the GPTMCTL register, the timer is enabled for event capture. When the selected input event is detected, the current timer counter value is captured in the GPTMTnR register and is available to be read by the microcontroller. The GPTM then asserts the CnERIS bit (and the CnEMIS bit, if the interrupt is not masked). The GPTMTnV contains the free-running value of the timer and can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

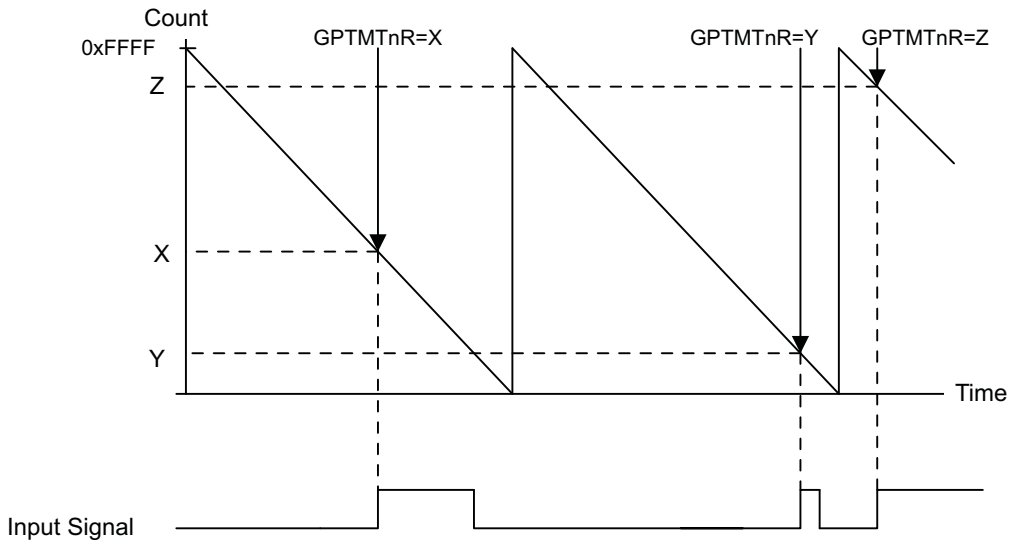
In addition to generating interrupts, a µDMA trigger can be generated. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See Channel Configuration in the *Micro Direct Memory Access (µDMA)* chapter.

After an event has been captured, the timer does not stop counting. It continues to count until the TnEN bit is cleared. When the timer reaches the timeout value, it is reloaded with the value from the GPTMTnILR register.

Figure 2-4 shows how input edge timing mode works. In the diagram, it is assumed that the start value of the timer is the default value of 0xFFFF, and the timer is configured to capture rising edge events.

Each time a rising edge event is detected, the current count value is loaded into the GPTMTnR register, and is held there until another rising edge is detected (at which point the new count value is loaded into the GPTMTnR register).

Figure 2-4. 16-Bit Input Edge-Time Mode Example



2.3.2.5 PWM Mode

NOTE: The prescaler is not available in 16-Bit PWM mode.

Copyright © 2012–2019, Texas Instruments Incorporated

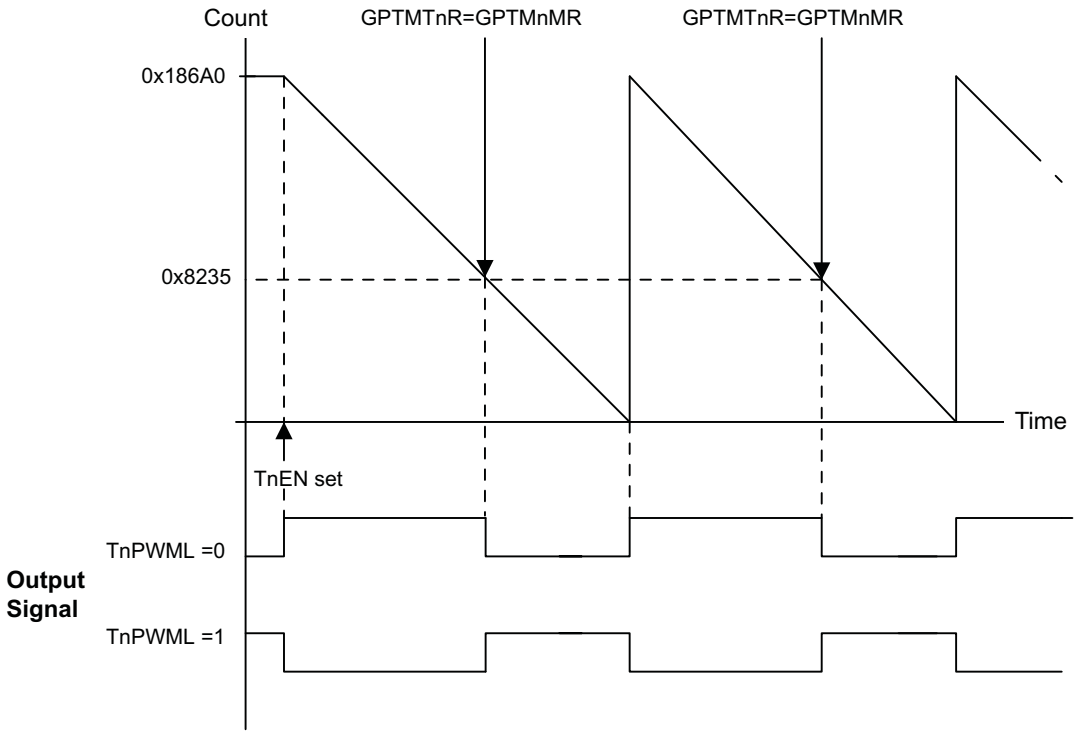
The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a 16-bit down-counter with a start value (and thus period) defined by the GPTMTnILR register. In this mode, the PWM frequency and period are synchronous events and therefore guaranteed to be glitch free. PWM mode is enabled with the GPTMTnMR register by setting the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2.

When software writes the TnEN bit in the GPTMCTL register, the counter begins counting down until it reaches the 0x0 state. On the next counter cycle, the counter reloads its start value from the GPTMTnILR register and continues counting until disabled by software clearing the TnEN bit in the GPTMCTL register. No interrupts or status bits are asserted in PWM mode.

The output PWM signal asserts when the counter is at the value of the GPTMTnILR register (its start state), and is deasserted when the counter value equals the value in the GPTMTnMATCHR register. Software has the capability of inverting the output PWM signal by setting the TnPWML bit in the GPTMCTL register.

Figure 2-5 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 100-MHz input clock and TnPWML =0 (duty cycle would be 33% for the TnPWML =1 configuration). For this example, the start value is GPTMTnILR=0x186A0 and the match value is GPTMTnMATCHR=0x8235.

Figure 2-5. 16-Bit PWM Mode Example



2.3.3 DMA Operation

The timers each have a dedicated µDMA channel and can provide a request signal to the µDMA controller. The request is a burst type and occurs whenever a timer raw interrupt condition occurs. The arbitration size of the µDMA transfer should be set to the amount of data that should be transferred whenever a timer event occurs.

For example, to transfer 256 items, 8 items at a time every 10 ms, configure a timer to generate a periodic timeout at 10 ms. Configure the µDMA transfer for a total of 256 items, with a burst size of 8 items. Each time the timer times out, the µDMA controller transfers 8 items, until all 256 items have been transferred.

No other special steps are needed to enable Timers for µDMA operation. Refer to the *Micro Direct Memory Access (µDMA)* chapter for more details about programming the µDMA controller.

Copyright © 2012–2019, Texas Instruments Incorporated

2.3.4 Accessing Concatenated Register Values

The GPTM is placed into concatenated mode by writing a 0x0 or a 0x1 to the GPTMCFG bit field in the GPTM Configuration (GPTMCFG) register. In both configurations, certain registers are concatenated to form pseudo 32-bit registers. These registers include:

- GPTM Timer A Interval Load (GPTMTAILR) register [15:0]
- GPTM Timer B Interval Load (GPTMTBILR) register [15:0]
- GPTM Timer A (GPTMTAR) register [15:0]
- GPTM Timer B (GPTMTBR) register [15:0]
- GPTM Timer A Value (GPTMTAV) register [15:0]
- GPTM Timer B Value (GPTMTBV) register [15:0]
- GPTM Timer A Match (GPTMTAMATCHR) register [15:0]
- GPTM Timer B Match (GPTMTBMATCHR) register [15:0]

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]

2.4 Initialization and Configuration

To use a GPTM, the appropriate TIMERn bit must be set in the RCGC1 register. If using any CCP pins, the clock to the appropriate GPIO module must be enabled via the RCGC1 register (see the System Control chapter).

To find out which GPIO port to enable, refer to the PMCn fields in the GPIOPCTL register to assign the CCP signals to the appropriate pins (see the GPIOs chapter).

This section shows module initialization and configuration examples for each of the supported timer modes.

2.4.1 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)
    1. Write a value of 0x1 for one-shot mode.
    2. 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).

Copyright © 2012–2019, Texas Instruments Incorporated

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.

2.4.2 Real-Time Clock (RTC) Mode

To use the RTC mode, the timer must have a 32.768-KHz input signal on an even CCP input. To enable the RTC feature, follow these steps:

1. Ensure the timer is disabled (the TAEN bit is cleared) before making any changes.
2. Write the GPTM Configuration Register (GPTMCFG) with a value of 0x0000.0001.
3. Write the match value to the GPTM Timer n Match Register (GPTMTnMATCHR).
4. Set/clear the RTCEN bit in the GPTM Control Register (GPTMCTL) as needed.
5. If interrupts are required, set the RTCIM bit in the GPTM Interrupt Mask Register (GPTMIMR).
6. Set the TAEN bit in the GPTMCTL register to enable the timer and start counting.

When the timer count equals the value in the GPTMTnMATCHR register, the GPTM asserts the RTCRIS bit in the GPTMRIS register and continues counting until Timer A is disabled or a hardware reset. The interrupt is cleared by writing the RTCCINT bit in the GPTMICR register.

2.4.3 Input Edge-Count Mode

A timer is configured to input edge-count mode by the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the GPTM Configuration (GPTMCFG) register with a value of 0x0000.0004.
3. In the GPTM Timer Mode (GPTMTnMR) register, write the TnCMR field to 0x0 and the TnMR field to 0x3.
4. Configure the type of event(s) that the timer captures by writing the TnEVENT field of the GPTM Control (GPTMCTL) register.
5. If a prescaler is to be used, write the prescale value to the GPTM Timer n Prescale Register (GPTMTnPR).
6. Load the timer start value into the GPTM Timer n Interval Load (GPTMTnILR) register.
7. Load the event count into the GPTM Timer n Match (GPTMTnMATCHR) register.
8. If interrupts are required, set the CnMIM bit in the GPTM Interrupt Mask (GPTMIMR) register.
9. Set the TnEN bit in the GPTMCTL register to enable the timer and begin waiting for edge events.
10. Poll the CnMRIS bit in the GPTMRIS register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the CnMCINT bit of the GPTM Interrupt Clear (GPTMICR) register.

When counting down in In input edge-count mode, the timer stops after the programmed number of edge events has been detected. To re-enable the timer, ensure that the TnEN bit is cleared and repeat step 4 through step 9.

2.4.4 Input Edge Timing Mode

A timer is configured to input edge timing mode by the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the GPTM Configuration (GPTMCFG) register with a value of 0x0000.0004.
3. In the GPTM Timer Mode (GPTMTnMR) register, write the TnCMR field to 0x1 and the TnMR field to 0x3.
4. Configure the type of event that the timer captures by writing the TnEVENT field of the GPTM Control (GPTMCTL) register.
5. Load the timer start value into the GPTM Timer n Interval Load (GPTMTnILR) register.
6. If interrupts are required, set the CnEIM bit in the GPTM Interrupt Mask (GPTMIMR) register.

Copyright © 2012–2019, Texas Instruments Incorporated

7. Set the TnEN bit in the GPTM Control (GPTMCTL) register to enable the timer and start counting.
8. Poll the CnERIS bit in the GPTMRIS register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the CnECINT bit of the GPTM Interrupt Clear (GPTMICR) register. The time at which the event happened can be obtained by reading the GPTM Timer n (GPTMTnR) register.

In Input Edge Timing mode, the timer continues running after an edge event has been detected, but the timer interval can be changed at any time by writing the GPTMTnILR register. The change takes effect at the next cycle after the write.

2.4.5 16-Bit PWM Mode

A timer is configured to PWM mode using the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes
2. Write the GPTM Configuration (GPTMCFG) register with a value of 0x0000.0004.
3. In the GPTM Timer Mode (GPTMTnMR) register, set the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the TnPWML field of the GPTM Control (GPTMCTL) register.
5. Load the timer start value into the GPTM Timer n Interval Load (GPTMTnILR) register.
6. Load the GPTM Timer n Match (GPTMTnMATCHR) register with the match value.
7. Set the TnEN bit in the GPTM Control (GPTMCTL) register to enable the timer and begin generation of the output PWM signal.

In PWM Timing mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing the GPTMTnILR register, and the change takes effect at the next cycle after the write.

2.5 Register Map

Table 2-4 lists the GPTM registers. The offset listed is a hexadecimal increment to the register 's address, relative to that timer's base address:

- Timer 0: 0x4003.0000
- Timer 1: 0x4003.1000
- Timer 2: 0x4003.2000
- Timer 3: 0x4003.3000

Note that the GP Timer module clock must be enabled before the registers can be programmed (see the RCGC1 register in the *System Control* chapter). Before any timer module registers are accessed, there must be a delay of three system clocks after the timer module clock is enabled.

Table 2-4. Timers Register Map

Offset	Name	Type	Reset	Description
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM Timer A Mode
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM Timer B Mode
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask
0x01C	GPTMRIS	R	0x0000.0000	GPTM Raw Interrupt Status
0x020	GPTMMIS	R	0x0000.0000	GPTM Masked Interrupt Status
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear
0x028	GPTMTAILR	R/W	0xFFFF.FFFF	GPTM Timer A Interval Load

Copyright © 2012–2019, Texas Instruments Incorporated

Table 2-4. Timers Register Map (continued)

Offset	Name	Type	Reset	Description
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM Timer B Interval Load
0x030	GPTMTAMATCHR	R/W	0xFFFF.FFFF	GPTM Timer A Match
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM Timer B Match
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM Timer A Prescale
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM Timer B Prescale
0x040	GPTMTAPMR	R/W	0x0000.0000	GPTM TimerA Prescale Match
0x044	GPTMTBPMR	R/W	0x0000.0000	GPTM TimerB Prescale Match
0x048	GPTMTAR	R	0xFFFF.FFFF	GPTM Timer A
0x04C	GPTMTBR	R	0x0000.FFFF	GPTM Timer B
0x050	GPTMTAV	R/W	0xFFFF.FFFF	GPTM Timer A Value
0x054	GPTMTBV	R/W	0x0000.FFFF	GPTM Timer B Value

2.6 Register Descriptions

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

2.6.1 GPTM Configuration (GPTMCFG) Register, offset 0x000

The GPTM Configuration (GPTMCFG) register configures the global operation of the GPTM module. The value written to this register determines whether the GPTM is in 32- or 16-bit mode.

Important: Bits in this register should only be changed when the TAEN and TBEN bits in the GPTMCTL register are cleared.

Figure 2-6. GPTM Configuration (GPTMCFG) Register

31	3	2	0
Reserved		GPTMCFG	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-5. GPTM Configuration (GPTMCFG) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
2-0	GPTMCFG		GPTM Configuration
		0x0	32-bit timer configuration
		0x1	32-bit real-time clock (RTC) counter configuration.
		0x2-0x3	Reserved
		0x4	16-bit timer configuration The function is controlled by bits 1:0 of GPTMTAMR and GPTMTBMR.
		0x5-0x7	Reserved

2.6.2 GPTM Timer A Mode (GPTMTAMR) Register, offset 0x004

The GPTM Timer A Mode (GPTMTAMR) 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 0x2.

Copyright © 2012–2019, Texas Instruments Incorporated

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.

Figure 2-7. GPTM Timer A Mode (GPTMTAMR) Register

31							8
Reserved							
7	6	5	4	3	2	1	0
TASNAPS	TAWOT	TAMIE	TACDIR	TAAMS	TACMR	TAMR	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-6. GPTM Timer A Mode (GPTMTAMR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7	TASNAPS		GPTM Timer A Snap-Shot Mode
		0	Snap-shot mode is disabled.
		1	If Timer A is configured in the periodic mode, the actual free-running value of Timer A is loaded at the time-out event into the GPTM Timer A (GPTMTAR) register.
6	TAWOT		GPTM Timer A Wait-on-Trigger
		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 2-2. This function is valid for both one-shot and periodic modes. This bit must be clear for GP Timer Module 0, Timer A.
5	TAMIE		GPTM Timer A Match Interrupt Enable
		0	The match interrupt is disabled
		1	An interrupt is generated when the match value in the GPTMTAMATCHR register is reached in the one-shot and periodic modes.
4	TACDIR		GPTM Timer A Count Direction
		0	The timer counts down
		1	When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0.
3	TAAMS		GPTM Timer A Alternate Mode Select
		0	Capture mode is enabled.
		0	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		GPTM Timer A Capture Mode
		0	Edge-Count mode
		1	Edge-Time mode
1-0	TAMR		GPTM Timer A Mode. The timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG register
		0x0	Reserved
		0x1	One-Shot Timer mode
		0x2	Periodic Timer mode
		0x3	Capture mode

Copyright © 2012–2019, Texas Instruments Incorporated

2.6.3 GPTM Timer B Mode (GPTMTBMR) Register, offset 0x008

The GPTM Timer B Mode (GPTMTBMR) 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 0x2.

This register controls the modes for Timer B when it is used individually. When Timer A and Timer B are concatenated, this register is ignored and GPTMTBMR controls the modes for both Timer A and Timer B.

Important: Bits in this register should only be changed when the TBEN bit in the GPTMCTL register is cleared.

Figure 2-8. GPTM Timer B Mode (GPTMTBMR) Register

31	30	29	28	27	26	25	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
TBSNAPS	TBWOT	TBMIE	TBCDIR	TBAMS	TBCMR	TBMR	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-7. GPTM Timer B Mode (GPTMTBMR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7	TBSNAPS		GPTM Timer B Snap-Shot Mode
		0	Snap-shot mode is disabled.
		1	If Timer B is configured in the periodic mode, the actual free-running value of Timer B is loaded at the time-out event into the GPTM Timer B (GPTMTBR) register.
6	TBWOT		GPTM Timer B Wait-on-Trigger
		0	Timer B begins counting as soon as it is enabled.
		1	If Timer B is enabled (TBEN is set in the GPTMCTL register), Timer B does not begin counting until it receives an it receives a trigger from the timer in the previous position in the daisy chain, see Figure 2-2. This function is valid for both one-shot and periodic modes.
5	TBMIE		GPTM Timer B Match Interrupt Enable
		0	The match interrupt is disabled.
		1	An interrupt is generated when the match value in the GPTMTBMATCHR register is reached in the one-shot and periodic modes.
4	TBCDIR		GPTM Timer B Count Direction
		0	The timer counts down.
		1	When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0.
3	TBAMS		GPTM Timer B Alternate Mode Select
		0	Capture 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		GPTM Timer B Capture Mode
		0	Edge-Count mode
		1	Edge-Time mode

Copyright © 2012–2019, Texas Instruments Incorporated

Table 2-7. GPTM Timer B Mode (GPTMTBMR) Register Field Descriptions (continued)

Bit	Field	Value	Description
1-0	TBMR		GPTM Timer B Mode. The timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG register.
		0x0	Reserved
		0x1	One-shot timer mode
		0x2	Periodic Ttimer mode
		0x3	Capture mode

2.6.4 GPTM Control (GPTMCTL) Register, offset 0x00C

The GPTM Control (GPTMCTL) 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.

Important: Bits in this register should only be changed when the TnEN bit for the respective timer is cleared.

Figure 2-9. GPTM Control (GPTMCTL) Register

31	30	29	28	27	26	25	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved	TBPWML	Reserved		TBEVENT		TBSTALL	TBEN
R-0	R/W-0	R-0		R/W-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	TAPWML	Reserved	RTCEN	TAEVENT		TASTALL	TAEN
R-0	R/W-0	R-0	R/W-0	R/W-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-8. GPTM Control (GPTMCTL) Register Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14	TBPWML		GPTM Timer B PWM Output Level
		0	Output is unaffected.
		1	Output is inverted.
13-12	Reserved		Reserved
11-10	TBEVENT		GPTM Timer B Event Mode
		0x0	Positive edge
		0x1	Negative edge
		0x2	Reserved
		0x3	Both edges
9	TBSTALL		GPTM Timer B Stall Enable
		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		GPTM Timer B Enable
		0	Timer B is disabled.
		1	Timer B is enabled and begins counting or the capture logic is enabled based on the GPTMCFG register.
7	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 2-8. GPTM Control (GPTMCTL) Register Field Descriptions (continued)

Bit	Field	Value	Description
6	TAPWML		GPTM Timer A PWM Output Level
		0	Output is unaffected.
		1	Output is inverted.
5	Reserved		Reserved
4	RTCEN		GPTM RTC Enable
		0	RTC counting is disabled.
		1	RTC counting is enabled.
3-2	TAEVENT		GPTM Timer A Event Mode
		0x0	Positive edge
		0x1	Negative edge
		0x2	Reserved
		0x3	Both edges
1	TASTALL		GPTM Timer A Stall Enable
		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
0	TAEN		GPTM Timer A Enable
		0	Timer A is disabled
		1	Timer A is enabled and begins counting or the capture logic is enabled based on the GPTMCFG register.

2.6.5 GPTM Interrupt Mask (GPTMIMR) Register, offset 0x018

The GPTM Interrupt Mask (GPTMIMR) register allows software to enable/disable GPTM controller-level interrupts. Setting a bit enables the corresponding interrupt, while clearing a bit disables it.

Figure 2-10. GPTM Interrupt Mask (GPTMIMR) Register

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

15	14	13	12	11	10	9	8
Reserved				TBMIM	CBEIM	CBMIM	TBTOIM
R-0				R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved			TAMIM	RTCIM	CAEIM	CAMIM	TATOIM
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-9. GPTM Interrupt Mask (GPTMIMR) Register Field Descriptions

Bit	Field	Value	Description
31-12			Reserved
11	TBMIM		GPTM Timer B Mode Match Interrupt Mask
		0	Interrupt is disabled.
		1	Interrupt is enabled.
10	CBEIM		GPTM Capture B Event Interrupt Mask
		0	Interrupt is disabled.
		1	Interrupt is enabled.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 2-9. GPTM Interrupt Mask (GPTMIMR) Register Field Descriptions (continued)

Bit	Field	Value	Description
9	CBMIM		GPTM Capture B Match Interrupt Mask
		0	Interrupt is disabled.
		1	Interrupt is enabled.
8	TBTOIM		GPTM Timer B Time-Out Interrupt Mask
		0	Interrupt is disabled.
		1	Interrupt is enabled.
7-5	Reserved		
4	TAMIM		GPTM Timer A Mode Match Interrupt Mask
		0	Interrupt is disabled.
		1	Interrupt is enabled.
3	RTCIM		GPTM RTC Interrupt Mask
		0	Interrupt is disabled.
		1	Interrupt is enabled.
2	CAEIM		GPTM Capture A Event Interrupt Mask
		0	Interrupt is disabled.
		1	Interrupt is enabled.
1	CAMIM		GPTM Capture A Match Interrupt Mask
		0	Interrupt is disabled.
		1	Interrupt is enabled.
0	TATOIM		GPTM Timer A Time-Out Interrupt Mask
		0	Interrupt is disabled.
		1	Interrupt is enabled.

2.6.6 GPTM Raw Interrupt Status (GPTMRIS) Register, offset 0x01C

The GPTM Raw Interrupt Status (GPTMRIS) 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.

Figure 2-11. GPTM Raw Interrupt Status (GPTMRIS) Register

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

15	14	13	12	11	10	9	8
Reserved				TBMRIS	CBERIS	CBMRIS	TBTORIS
R-0				R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
Reserved			TAMRIS	RTCRIS	CAERIS	CAMRIS	TATORIS
R-0			R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-10. GPTM Raw Interrupt Status (GPTMRIS) Register Field Descriptions

Bit	Field	Value	Description
31-12			Reserved
11	TBMRIS		GPTM Timer B Mode Match Raw Interrupt
		0	The match value has not been reached
		1	The TBMIE bit is set in the GPTMTBMR register, and the match value in the GPTMTBMATCHR register has been reached when in the one-shot and periodic modes.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 2-10. GPTM Raw Interrupt Status (GPTMRIS) Register Field Descriptions (continued)

Bit	Field	Value	Description
10	CBERIS		GPTM Capture B Event Raw Interrupt
		0	The Capture B event has not occurred.
		1	The Capture B event has occurred.
9	CBMRIS		GPTM Capture B Match Raw Interrupt
		0	The Capture B match has not occurred.
		1	The Capture B match has occurred. This bit is cleared by writing a 1 to the CBMCINT bit in the GPTMICR register
8	TBTORIS		GPTM Timer B Time-Out Raw Interrupt
		0	Timer B has not timed out.
		1	ITimer B has timed out.
7-5	Reserved		
4	TAMRIS		GPTM Timer A Mode Match Raw Interrupt
		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 register has been reached when in the one-shot and periodic modes. This bit is cleared by writing a 1 to the TAMCINT bit in the GPTMICR register
3	RTCRIS		GPTM RTC Raw Interrupt
		0	The RTC event has not occurred
		1	The RTC event has occurred
2	CAERIS		GPTM Capture A Event Raw Interrupt
		0	The Capture A event has not occurred.
		1	The Capture A event has occurred.
1	CAMRIS		GPTM Capture A Match Raw Interrupt
		0	The Capture A match has not occurred.
		1	The Capture A match has occurred. This bit is cleared by writing a 1 to the CAMCINT bit in the GPTMICR register.
0	TATORIS		GPTM Timer A Time-Out Raw Interrupt
		0	Timer A has not timed out.
		1	Timer A has timed out. This bit is cleared by writing a 1 to the TATOCINT bit in the GPTMICR register.

2.6.7 GPTM Masked Interrupt Status (GPTMMIS) Register, offset 0x020

The GPTM Masked Interrupt Status (GPTMMIS) register shows 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.

Figure 2-12. GPTM Masked Interrupt Status (GPTMMIS) Register

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

15	14	13	12	11	10	9	8
Reserved				TBMMIS	CBEMIS	CBMMIS	TBTOMIS
R-0				R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
Reserved			TAMMIS	RTCMIS	CAEMIS	CAMMIS	TATOMIS
R-0			R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 2-11. GPTM Masked Interrupt Status (GPTMMIS) Register Field Descriptions

Bit	Field	Value	Description
31-12			Reserved
11	TBMMIS		GPTM Timer B Mode Match Masked Interrupt
		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.
10	CBEMIS		GPTM Capture B Event Masked Interrupt
		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		GPTM Capture B Match Masked Interrupt
		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		GPTM Timer B Time-Out Masked Interrupt
		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		
4	TAMMIS		GPTM Timer A Mode Match Raw Interrupt
		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		GPTM RTC Masked Interrupt
		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	CAEMIS		GPTM Capture A Event Masked Interrupt
		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		GPTM Capture A Match Masked Interrupt
		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		GPTM Timer A Time-Out Masked Interrupt
		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.

2.6.8 GPTM Interrupt Clear (GPTMICR) Register, offset 0x024

The GPTM Interrupt Clear (GPTMICR) 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.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 2-13. GPTM Interrupt Clear (GPTMICR) Register

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

15	14	13	12	11	10	9	8
Reserved				TBMCINT	CBECINT	CBMCINT	TBTOCINT
R-0				W1C-0	W1C-0	W1C-0	W1C-0

7	6	5	4	3	2	1	0
Reserved			TAMCINT	RTCCINT	CAECINT	CAMCINT	TATOCINT
R-0			W1C-0	W1C-0	W1C-0	W1C-0	W1C-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-12. GPTM Interrupt Clear (GPTMICR) Register Field Descriptions

Bit	Field	Value	Description
31-12	Reserved		Reserved
11	TBMCINT		GPTM Timer B Mode Match Interrupt Clear. Writing a 1 to this bit clears the TBMRIS bit in the GPTMRIS register and the TBMMIS bit in the GPTMMIS register.
10	CBECINT		GPTM Capture B Event Interrupt Clear Writing a 1 to this bit clears the CBERIS bit in the GPTMRIS register and the CBEMIS bit in the GPTMMIS register.
9	CBMCINT		GPTM Capture B Match Interrupt Clear Writing a 1 to this bit clears the CBMRIS bit in the GPTMRIS register and the CBMMIS bit in the GPTMMIS register.
8	TBTOCINT		GPTM Timer B Time-Out Interrupt Clear Writing a 1 to this bit clears the TBTORIS bit in the GPTMRIS register and the TBTOMIS bit in the GPTMMIS register.
7-5	Reserved		Reserved
4	TAMCINT		GPTM Timer A Mode Match Interrupt Clear Writing a 1 to this bit clears the TAMRIS bit in the GPTMRIS register and the TAMMIS bit in the GPTMMIS register.
3	RTCCINT		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		GPTM Capture A Event Interrupt Clear Writing a 1 to this bit clears the CAERIS bit in the GPTMRIS register and the CAEMIS bit in the GPTMMIS register.
1	CAMCINT		GPTM Capture A Match Interrupt Clear Writing a 1 to this bit clears the CAMRIS bit in the GPTMRIS register and the CAMMIS bit in the GPTMMIS register.
0	TATOCINT		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.

2.6.9 GPTM Timer A Interval Load (GPTMTAILR) Register, offset 0x028

The GPTM Timer A Interval Load (GPTMTAILR) register is used to load the starting count value into the timer, when the timer is counting down. When the timer is counting up, this register sets the upper bound for the timeout event.

When a GPTM is configured to one of the 32-bit modes, GPTMTAILR appears as a 32-bit register (the upper 16-bits correspond to the contents of the GPTM Timer B Interval Load (GPTMTBILR) register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of GPTMTBILR.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 2-14. GPTM Timer A Interval Load (GPTMTAILR) Register

31	0
TAILR	
R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-13. GPTM Timer A Interval Load (GPTMTAILR) Register Field Descriptions

Bit	Field	Value	Description
31-0	TAILR	0x0000. FFFF	GPTM Timer A Interval Load Register Writing this field loads the counter for Timer A. A read returns the current value of GPTMTAILR.

2.6.10 GPTM Timer B Interval Load (GPTMTBILR) Register, offset 0x02C

The GPTM Timer B Interval Load (GPTMTBILR) register is used to load the starting count value into the timer, when the timer is counting down . When the timer is counting up, this register sets the upper bound for the timeout event.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the GPTMTAILR register. Reads from this register return the current value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the load value. Bits 31:16 are reserved in both cases.

Figure 2-15. GPTM Timer A Interval Load (GPTMTBILR) Register

31	16	15	0
Reserved		TBILR	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-14. GPTM Timer A Interval Load (GPTMTBILR) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	TBILR	0x0000. FFFF	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 GPTM is in 32-bit mode, writes are ignored, and reads return the current value of GPTMTBILR.

2.6.11 GPTM Timer A Match (GPTMTAMATCHR) Register, offset 0x030

The GPTM Timer A Match (GPTMTAMATCHR) register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode.

In Edge-Count mode, this register along with GPTMTAILR, determines how many edge events are counted. The total number of edge events counted is equal to the value in GPTMTAILR minus this value.

In PWM mode, this value along with GPTMTAILR, determines the duty cycle of the output PWM signal.

When a GPTM is configured to one of the 32-bit modes, GPTMTAMATCHR appears as a 32-bit register (the upper 16-bits correspond to the contents of the GPTM Timer B Match (GPTMTBMATCHR) register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of GPTMTBMATCHR.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 2-16. GPTM Timer A Match (GPTMTAMATCHR) Register

31	0
TAMR	
R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-15. GPTM Timer A Match (GPTMTAMATCHR) Register Field Descriptions

Bit	Field	Value	Description
31-0	TAMR	0x0000. FFFF	GPTM Timer A Match Register This value is compared to the GPTMTAR register to determine match events.

2.6.12 GPTM Timer B Match (GPTMTBMATCHR) Register, offset 0x034

The GPTM Timer B Match (GPTMTBMATCHR) register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode.

In Edge-Count mode, this register along with GPTMTBILR, determines how many edge events are counted. The total number of edge events counted is equal to the value in GPTMTBILR minus this value.

In PWM mode, this value along with GPTMTBILR, determines the duty cycle of the output PWM signal.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the GPTMTAMATCHR register. Reads from this register return the current match value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the match value. Bits 31:16 are reserved in both cases.

Figure 2-17. GPTM Timer B Match (GPTMTBMATCHR) Register

31	16	15	0
Reserved		TBMR	
R-0		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-16. GPTM Timer B Match (GPTMTBMATCHR) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	TBMR	0x0000. FFFF	GPTM Timer B Match Register This value is compared to the GPTMTBR register to determine match events.

2.6.13 GPTM Timer A Prescale (GPTMTAPR) Register, offset 0x038

The GPTM Timer A Prescale (GPTMTAPR) register allows software to extend the range of the 16-bit timers in periodic and one-shot modes. In Edge-Count mode, this register is the MSB of the 24-bit count value.

Figure 2-18. GPTM Timer A Prescale (GPTMTAPR) Register

31	8	7	0
Reserved		TAPSR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 2-17. GPTM Timer A Prescale (GPTMTAPR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	TAPSR	0x00	GPTM Timer A Prescale The register loads this value on a write. A read returns the current value of the register. Refer to Table 2-2 for more details and an example.

2.6.14 GPTM Timer B Prescale (GPTMTBPR) Register, offset 0x03C

The GPTM Timer B Prescale (GPTMTBPR) register allows software to extend the range of the 16-bit timers in periodic and one-shot modes. In Edge-Count mode, this register is the MSB of the 24-bit count value.

Figure 2-19. GPTM Timer A Prescale (GPTMTBPR) Register

31	8	7	0
Reserved		TBPSR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 2-18. GPTM Timer A Prescale (GPTMTBPR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	TBPSR	0x00	GPTM Timer B Prescale The register loads this value on a write. A read returns the current value of the register. Refer to Table 2-2 for more details and an example.

2.6.15 GPTM Timer A Prescale Match (GPTMTAPMR) Register, offset 0x040

The GPTM Timer A Prescale Match (GPTMTAPMR) register effectively extends the range of GPTMTAMATCHR to 24 bits when operating in 16-bit one-shot or periodic mode.

Figure 2-20. GPTM Timer A Prescale Match (GPTMTAPMR) Register

31	8	7	0
Reserved		TAPSMR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 2-19. GPTM Timer A Prescale Match (GPTMTAPMR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	TAPSMR	0x00	GPTM Timer A Prescale Match This value is used alongside GPTMTAMATCHR to detect timer match events while using a prescaler.

2.6.16 GPTM Timer B Prescale Match (GPTMTBPMR) Register, offset 0x044

The GPTM Timer B Prescale Match (GPTMTBPMR) register effectively extends the range of GPTMTBMATCHR to 24 bits when operating in 16-bit one-shot or periodic mode.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 2-21. GPTM Timer B Prescale Match (GPTMTBPMR) Register

31	8	7	0
Reserved		TBPSMR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-20. GPTM Timer B Prescale Match (GPTMTBPMR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	TBPSMR	0x00	GPTM Timer B Prescale Match This value is used alongside GPTMTBMATCHR to detect timer match events while using a prescaler.

2.6.17 GPTM Timer A (GPTMTAR) Register, offset 0x048

The GPTM Timer A (GPTMTAR) register shows the current value of the Timer A counter in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place. Also in Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, GPTMTAR appears as a 32-bit register (the upper 16-bits correspond to the contents of the GPTM Timer B (GPTMTBR) register). In the16-bit Input Edge Count, Input Edge Time, and PWM modes, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler, which is the upper 8 bits of the count. Bits 31:24 always read as 0. To read the value of the prescaler in 16-bit One-Shot and Periodic modes, read bits [23:16] in the GPTMTAV register.

Figure 2-22. GPTM Timer A (GPTMTAR) Register

31	0
TAR	
R-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-21. GPTM Timer A (GPTMTAR) Register Field Descriptions

Bit	Field	Value	Description
31-0	TAR	0xFFFF .FFFF	GPTM TimerA 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.

2.6.18 GPTM Timer B (GPTMTBR) Register, offset 0x04C

The GPTM Timer B (GPTMTBR) register shows the current value of the Timer B counter in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place. Also in Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the GPTMTAR register. Reads from this register return the current value of Timer B. In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler in Input Edge Count, Input Edge Time, and PWM modes, which is the upper 8 bits of the count. Bits 31:24 are reserved in both cases.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 2-23. GPTM Timer B (GPTMTBR) Register

31	16	15	0
Reserved		TBR	
R-0		R-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-22. GPTM Timer B (GPTMTBR) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	TBR	0xFFFF.FFFF	GPTM TimerB 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.

2.6.19 GPTM Timer A Value (GPTMTAV) Register, offset 0x050

When read, the GPTM Timer A Value (GPTMTAV) register shows the current, free-running value of Timer A in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry. When written, the value written into this register is loaded into the GPTMTAR register on the next clock cycle. In Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, GPTMTAV appears as a 32-bit register (the upper 16-bits correspond to the contents of the GPTM Timer B Value (GPTMTBV) register). In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count. Bits 31:24 always read as 0.

NOTE: The GPTMTAV register cannot be written in edge-count mode.

Figure 2-24. GPTM Timer A Value (GPTMTAV) Register

31	0
TAV	
R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-23. GPTM Timer A Value (GPTMTAV) Register Field Descriptions

Bit	Field	Value	Description
31-0	TAV	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.

2.6.20 GPTM Timer B Value (GPTMTBV) Register, offset 0x054

When read, the GPTM Timer B Value (GPTMTBV) register shows the current, free-running value of Timer B in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry. When written, the value written into this register is loaded into the GPTMTBR register on the next clock cycle. In Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the GPTMTAV register. Reads from this register return the current free-running value of Timer B. In a 16-bit mode, bits 15:0 contain the current, free-running value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count. Bits 31:24 are reserved in both cases.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 2-25. GPTM Timer B Value (GPTMTBV) Register

31	16	15	0
Reserved		TBV	
		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-24. GPTM Timer B Value (GPTMTBV) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	TBV	0xFFFF .FFFF	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.

Copyright © 2012–2019, Texas Instruments Incorporated

M3 Watchdog Timers

This chapter discusses the functions of the Watchdog Timer 0 and Watchdog Timer 1 modules. A watchdog timer can generate an interrupt or a reset when a time-out value is reached.

Topic **Page**

3.1 Introduction **325**
3.2 Register Map **326**
3.3 Register Descriptions **327**

Copyright © 2012–2019, Texas Instruments Incorporated

3.1 Introduction

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 M3 Subsystem has two watchdog timer modules; one module is clocked by the system clock (Watchdog Timer 0) and the other is clocked by the OSCCLK (Watchdog Timer 1). The two modules are identical except that WDT1 is in a different clock domain, and therefore requires synchronizers. As a result, WDT1 has a bit defined in the watchdog timer control (WDTCTL) register to indicate when a write to a WDT1 register is complete. Software can use this bit to ensure that the previous access has completed before starting the next access.

The watchdog timer modules include the following features:

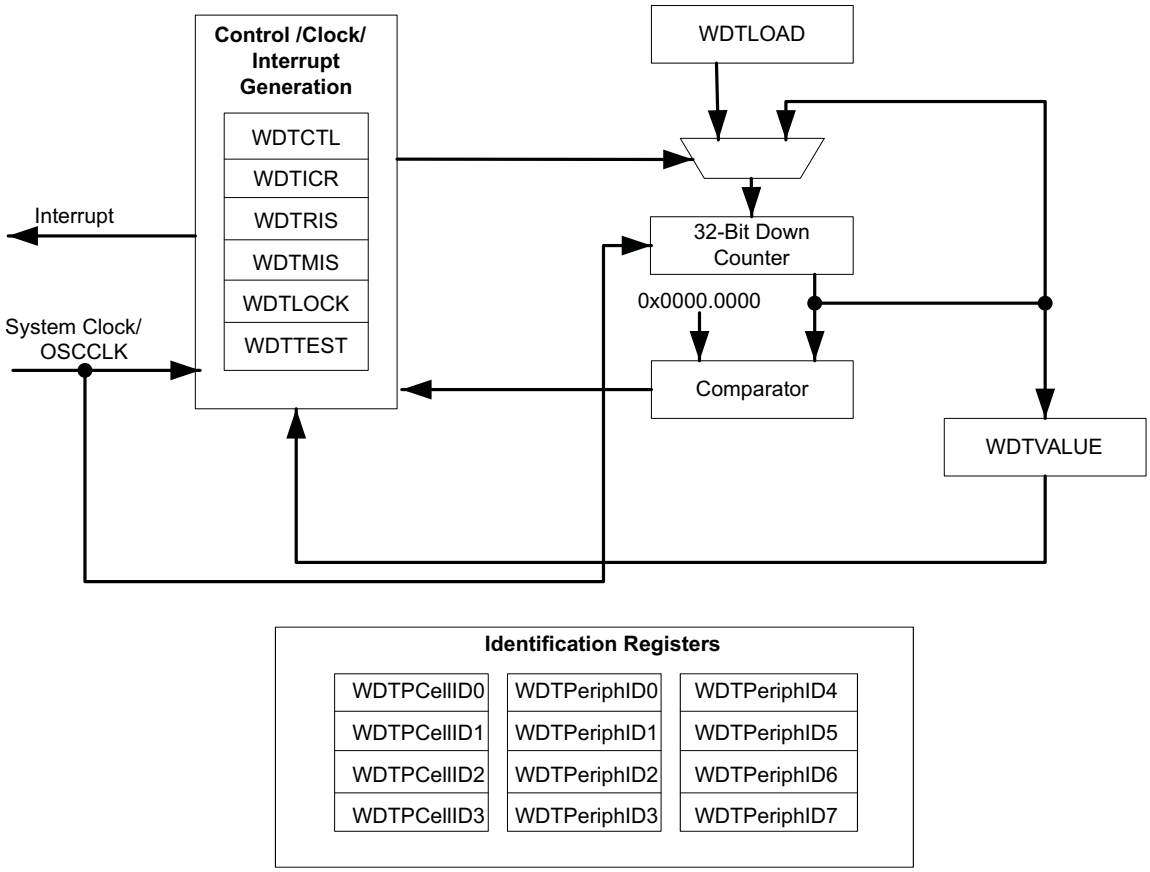
- 32-bit down counter with a programmable load register
- Separate watchdog clock with an enable
- Programmable interrupt generation logic with interrupt masking
- Lock register protection from runaway software
- Reset generation logic with an enable/disable
- User-enabled stalling when the microcontroller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

The watchdog timer can be configured to generate an interrupt to the controller on its first time-out, and to generate a reset signal on its second time-out. Once the watchdog timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

3.1.1 Block Diagram

The watchdog timer module block diagram is shown in Figure 3-1.

Figure 3-1. Watchdog Timer Module Block Diagram



Copyright © 2012–2019, Texas Instruments Incorporated

3.1.2 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. After the first time-out event, the 32-bit counter is re-loaded with the value of the Watchdog Timer Load (WDTLOAD) register, and the timer resumes counting down from that value. Once the Watchdog Timer has been configured, the Watchdog Timer Lock (WDTLOCK) register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled by setting the RESEN bit in the WDTCTL register, the watchdog timer asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the WDTLOAD register, and counting resumes from that value.

If WDTLOAD is written with a new value while the watchdog timer counter is counting, then the counter is loaded with the new value and continues counting.

Writing to WDTLOAD does not clear an active interrupt. An interrupt must be specifically cleared by writing to the Watchdog Interrupt Clear (WDTICR) register.

The watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is re-enabled, the 32-bit counter is preloaded with the load register value and not its last state.

3.1.2.1 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.

3.1.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the WDT bit in the RCGC0 register. See the System Control chapter, *Run Mode Clock Gating Control Register 0 (RCGC0)* section.

The watchdog timer is configured using the following sequence:

1. Load the WDTLOAD register with the desired timer load value.
2. If WDT1, wait for the WRC bit in the WDTCTL register to be set.
3. If the Watchdog is configured to trigger system resets, set the RESEN bit in the WDTCTL register.
4. If WDT1, wait for the WRC bit in the WDTCTL register to be set.
5. Set the INTEN bit in the WDTCTL register to enable the Watchdog and lock the control register.

If software requires that all of the watchdog registers are locked, the watchdog timer module can be fully locked by writing any value to the WDTLOCK register. To unlock the watchdog timer, write a value of 0x1ACC.E551.

3.2 Register Map

Table 3-1 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 (ending address of 0x4000.0FFF)
- WDT1: 0x4000.1000 (ending address of 0x4000.1FFF)

Note that the watchdog timer module clock must be enabled before the registers can be programmed. See the System Control chapter, *Run Mode Clock Gating Control Register 0 (RCGC0)* section.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 3-1. Watchdog Timers Register Map

Offset	Name	Type	Reset	Description
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load
0x004	WDTVALUE	R	0xFFFF.FFFF	Watchdog Value
0x008	WDTCTL	R/W	0x0000.0000 (WDT0) 0x8000.0000 (WDT1)	Watchdog Control
0x00C	WDTICR	W	-	Watchdog Interrupt Clear
0x010	WDTRIS	R	0x0000.0000	Watchdog Raw Interrupt Status
0x014	WDTMIS	R	0x0000.0000	Watchdog Masked Interrupt Status
0x018	WDTTEST	R/W	0x0000.0000	Watchdog Test
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock
0xFD0	WDTPeriphID4	R	0x0000.0000	Watchdog Peripheral Identification 4
0xFD4	WDTPeriphID5	R	0x0000.0000	Watchdog Peripheral Identification 5
0xFD8	WDTPeriphID6	R	0x0000.0000	Watchdog Peripheral Identification 6
0xFDC	WDTPeriphID7	R	0x0000.0000	Watchdog Peripheral Identification 7
0xFE0	WDTPeriphID0	R	0x0000.0005	Watchdog Peripheral Identification 0
0xFE4	WDTPeriphID1	R	0x0000.0018	Watchdog Peripheral Identification 1
0xFE8	WDTPeriphID2	R	0x0000.0018	Watchdog Peripheral Identification 2
0xFEC	WDTPeriphID3	R	0x0000.0001	Watchdog Peripheral Identification 3
0xFF0	WDTPCellID0	R	0x0000.000D	Watchdog PrimeCell Identification 0
0xFF4	WDTPCellID1	R	0x0000.00F0	Watchdog PrimeCell Identification 1
0xFF8	WDTPCellID2	R	0x0000.0006	Watchdog PrimeCell Identification 2
0xFFC	WDTPCellID3	R	0x0000.00B1	Watchdog PrimeCell Identification 3

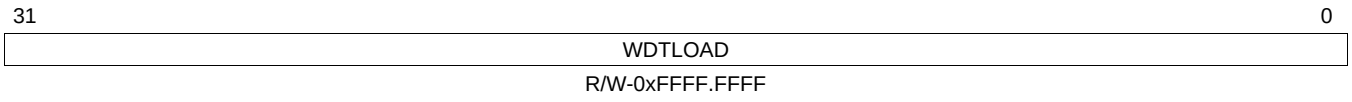
3.3 Register Descriptions

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

3.3.1 Watchdog Load (WDTLOAD) Register, offset 0x000

The watchdog load register (WDTLOAD) 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.

Figure 3-2. Watchdog Load (WDTLOAD) Register



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 3-2. Watchdog Load (WDTLOAD) Register Field Descriptions

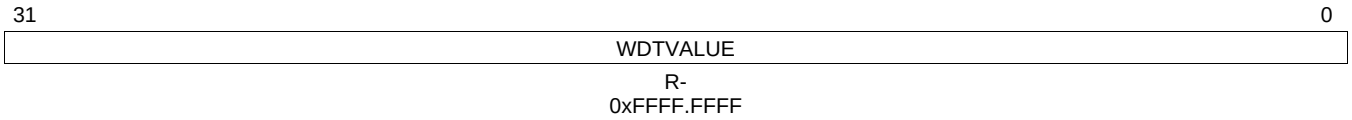
Bit	Field	Value	Description
31-0	WDTLOAD		Watchdog load value
		0	Register not written. Value not loaded.
		1	Register written. Vlaue loaded and counter restarts.

Copyright © 2012–2019, Texas Instruments Incorporated

3.3.2 Watchdog Value (WDTVALUE) Register, offset 0x004

The watchdog value (WDTVALUE) register contains the current count value of the timer.

Figure 3-3. Watchdog Value (WDTVALUE) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-3. Watchdog Value (WDTVALUE) Register Field Descriptions

Bit	Field	Value	Description
31-0	WDTVALUE		Watchdog value. Current value of the 32-bit down counter.

Copyright © 2012–2019, Texas Instruments Incorporated

3.3.3 Watchdog Control (WDTCTL) Register, offset 0x008

The watchdog timer can be configured to generate a reset signal (on second time-out) or an interrupt on time-out. When the watchdog interrupt has been enabled, all subsequent writes to the control register are ignored. The only mechanism that can re-enable writes is a hardware reset.

NOTE: 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.

Figure 3-4. Watchdog Control (WDTCTL) Register

31	30	2	1	0
WRC	Reserved		RESEN	ITEN
R-1	R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-4. Watchdog Control (WDTCTL) Register Field Descriptions

Bit	Field	Value	Description
31	WRC		Write compatible.
		0	A write access to one of the WDT1 registers is in progress.
		1	A write access is not in progress, and WDT1 registers can be read or written. **Note:** This bit is reserved for WDT0 and has a reset value of 0.
30-2	Reserved		Reserved
1	RESEN		Watchdog reset enable
		0	Disabled
		1	Enable the watchdog module reset output.
0	INTEN		Watchdog interrupt enable
		0	Interrupt event disabled (once this bit is set, it can only be cleared by a hardware reset).
		1	Interrupt event enabled. Once enabled, all writes are ignored.

3.3.4 Watchdog Interrupt Clear (WDTICR) Register, offset 0x00C

The Wwatchdog interrupt clear (WDTICR) register is the interrupt clear register. A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the WDTLOAD register. Value for a read or reset is indeterminate.

Figure 3-5. Watchdog Interrupt Clear (WDTICR) Register

31	0
WDTINTCLR	
W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-5. Watchdog Interrupt Clear (WDTICR) Register Field Descriptions

Bit	Field	Value	Description
31-0	WDTINTCLR		Watchdog interrupt clear

Copyright © 2012–2019, Texas Instruments Incorporated

3.3.5 Watchdog Raw Interrupt Status (WDTRIS) Register, offset 0x010

The watchdog raw interrupt status (WDTRIS) register is the raw interrupt status register. Watchdog interrupt events can be monitored via this register if the controller interrupt is masked.

Figure 3-6. Watchdog Raw Interrupt Status (WDTRIS) Register

31	1	0
Reserved		WDTRIS
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-6. Watchdog Raw Interrupt Status (WDTRIS) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	WDTRIS		Watchdog raw interrupt status
		0	The watchdog has not timed out.
		1	A watchdog time-out event has occurred.

3.3.6 Watchdog Masked Interrupt Status (WDTMIS) Register, offset 0x014

The watchdog masked interrupt status (WDTMIS) 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.

Figure 3-7. Watchdog Masked Interrupt Status (WDTMIS) Register

31	1	0
Reserved		WDTMIS
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-7. Watchdog Masked Interrupt Status (WDTMIS) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	WDTMIS		Watchdog masked interrupt status
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

3.3.7 Watchdog Test (WDTTEST) Register, offset 0x418

The watchdog test (WDTTEST) register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

Figure 3-8. Watchdog Test (WDTTEST) Register

31	9	8	7	0
Reserved		STALL	Reserved	
R-0		R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-8. Watchdog Test (WDTTEST) Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	STALL		Watchdog stall enable
		0	If the microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting.
		1	The watchdog timer continues counting if the microcontroller is stopped with a debugger.
7-0	Reserved		Reserved

3.3.8 Watchdog Lock (WDTLOCK) Register, offset 0xC00

Writing 0x1ACC.E551 to the watchdog lock (WDTLOCK) register enables write access to all other registers. Writing any other value to the WDTLOCK register re-enables the locked state for register writes to all the other registers. Reading the WDTLOCK register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the WDTLOCK register returns 0x0000.0001 (when locked; otherwise, the returned value is 0x0000.0000 (unlocked)).

Figure 3-9. Watchdog Lock (WDTLOCK) Register

31	0
WDTLOCK	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-9. Watchdog Lock (WDTLOCK) Register Field Descriptions

Bit	Field	Value	Description
31-0	Reserved		Watchdog lock
0	WDTMIS		A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates. A read of this register returns the following values: A watchdog time-out event has been signalled to the interrupt controller.
		0x0000.0001	Locked
		0x0000.0000	Unlocked

Copyright © 2012–2019, Texas Instruments Incorporated

3.3.9 Watchdog Peripheral Identification 4 (WDTPeriphID4) Register, offset 0xFD0

The watchdog peripheral identification (WDTPeriphIDn) registers are hard-coded and the fields within the register determine the reset value.

Figure 3-10. Watchdog Peripheral Identification 4 (WDTPeriphID4) Register

31	8	7	0
Reserved		PID4	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-10. Watchdog Peripheral Identification 4 (WDTPeriphID4) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID4		WDT Peripheral ID Register [7:0]

3.3.10 Watchdog Peripheral Identification 5 (WDTPeriphID5) Register, offset 0xFD4

The watchdog peripheral identification (WDTPeriphIDn) registers are hard-coded and the fields within the register determine the reset value.

Figure 3-11. Watchdog Peripheral Identification 5 (WDTPeriphID5) Register

31	8	7	0
Reserved		PID5	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-11. Watchdog Peripheral Identification 5 (WDTPeriphID5) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID5		WDT Peripheral ID Register [15:8]

3.3.11 Watchdog Peripheral Identification 6 (WDTPeriphID6) Register, offset 0xFD8

The watchdog peripheral identification (WDTPeriphIDn) registers are hard-coded and the fields within the register determine the reset value.

Figure 3-12. Watchdog Peripheral Identification 6 (WDTPeriphID6) Register

31	8	7	0
Reserved		PID6	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-12. Watchdog Peripheral Identification 6 (WDTPeriphID6) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID6		WDT Peripheral ID Register [23:16]

Copyright © 2012–2019, Texas Instruments Incorporated

3.3.12 Watchdog Peripheral Identification 7 (WDTPeriphID7) Register, offset 0xFDC

The watchdog peripheral identification (WDTPeriphIDn) registers are hard-coded and the fields within the register determine the reset value.

Figure 3-13. Watchdog Peripheral Identification 7 (WDTPeriphID7) Register

31	8	7	0
Reserved		PID7	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-13. Watchdog Peripheral Identification 7 (WDTPeriphID7) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID7		WDT Peripheral ID Register [31:24]

3.3.13 Watchdog Peripheral Identification 0 (WDTPeriphID0) Register, offset 0xFE0

The watchdog peripheral identification (WDTPeriphIDn) registers are hard-coded and the fields within the register determine the reset value.

Figure 3-14. Watchdog Peripheral Identification 0 (WDTPeriphID0) Register

31	8	7	0
Reserved		PID0	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-14. Watchdog Peripheral Identification 0 (WDTPeriphID0) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID0		WDT Peripheral ID Register [7:0]

3.3.14 Watchdog Peripheral Identification 1 (WDTPeriphID1) Register, offset 0xFE4

The watchdog peripheral identification (WDTPeriphIDn) registers are hard-coded and the fields within the register determine the reset value.

Figure 3-15. Watchdog Peripheral Identification 1 (WDTPeriphID1) Register

31	8	7	0
Reserved		PID1	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-15. Watchdog Peripheral Identification 1 (WDTPeriphID1) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID1		WDT Peripheral ID Register [15:8]

Copyright © 2012–2019, Texas Instruments Incorporated

3.3.15 Watchdog Peripheral Identification 2 (WDTPeriphID2) Register, offset 0xFE8

The watchdog peripheral identification (WDTPeriphIDn) registers are hard-coded and the fields within the register determine the reset value.

Figure 3-16. Watchdog Peripheral Identification 2 (WDTPeriphID2) Register

31	8	7	0
Reserved		PID2	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-16. Watchdog Peripheral Identification 2 (WDTPeriphID2) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID2		WDT Peripheral ID Register [23:16]

3.3.16 Watchdog Peripheral Identification 3 (WDTPeriphID3) Register, offset 0xFEC

The watchdog peripheral identification (WDTPeriphIDn) registers are hard-coded and the fields within the register determine the reset value.

Figure 3-17. Watchdog Peripheral Identification 3 (WDTPeriphID3) Register

31	8	7	0
Reserved		PID3	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-17. Watchdog Peripheral Identification 3 (WDTPeriphID3) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID3		WDT Peripheral ID Register [31:24]

3.3.17 Watchdog PrimeCell Identification 0 (WDTPCellID0) Register, offset 0xFF0

The watchdog primecell identification (WDTPCellIDn) registers are hard-coded and the fields within the register determine the reset value.

Figure 3-18. Watchdog PrimeCell Identification 0 (WDTPCellID0) Register

31	8	7	0
Reserved		CID0	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-18. Watchdog PrimeCell Identification 0 (WDTPCellID0) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID0		Watchdog PrimeCell ID Register [7:0]

Copyright © 2012–2019, Texas Instruments Incorporated

3.3.18 Watchdog PrimeCell Identification 1 (WDTPCellID1) Register, offset 0xFF4

The watchdog primecell identification (WDTPCellIDn) registers are hard-coded and the fields within the register determine the reset value.

Figure 3-19. Watchdog PrimeCell Identification 1 (WDTPCellID1) Register

31	8	7	0
Reserved		CID1	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-19. Watchdog PrimeCell Identification 1 (WDTPCellID1) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID1		Watchdog PrimeCell ID Register [15:8]

3.3.19 Watchdog PrimeCell Identification 2 (WDTPCellID2) Register, offset 0xFF8

Thewatchdog primecell identification (WDTPCellIDn)registers are hard-coded and the fields within the register determine the reset value.

Figure 3-20. Watchdog PrimeCell Identification 2 (WDTPCellID2) Register

31	8	7	0
Reserved		CID2	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-20. Watchdog PrimeCell Identification 2 (WDTPCellID2) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID2		Watchdog PrimeCell ID Register [23:16]

3.3.20 Watchdog PrimeCell Identification 3 (WDTPCellID3) Register, offset 0xFFC

The watchdog primecell identification (WDTPCellIDn)registers are hard-coded and the fields within the register determine the reset value.

Figure 3-21. Watchdog PrimeCell Identification 3 (WDTPCellID3) Register

31	8	7	0
Reserved		CID3	
RO		RO	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-21. Watchdog PrimeCell Identification 3 (WDTPCellID3) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID3		Watchdog PrimeCell ID Register [31:24]

Copyright © 2012–2019, Texas Instruments Incorporated

General-Purpose Input/Output (GPIO)

This chapter describes the M3 GPIO and C28 GPIO functionality and GPIO mux configurations.

After a reset to the device, the M3 core controls all GPIOs except GPIO128 - GPIO135. The C28 core always has control of GPIO128 - GPIO135. The GPIOCSELx register can be used to give the C28 core control of any GPIO. If the M3 core has control of a GPIO, the M3 muxing options can be used, and if the C28 core has control of a GPIO, the C28 muxing options can be used. Regardless of which core owns a GPIO, the M3 core always controls the pull-down functionality of each GPIO except GPIO128 – GPIO135. The C28 core controls the pull-down functionality of GPIO128 – GPIO135.

See the General-Purpose Input/Output (GPIO) and C28 General-Purpose Input/Output (GPIO) sections of this chapter for more information.

Topic **Page**

4.1 General-Purpose Input/Output (GPIO) **337**
4.2 C28 General-Purpose Input/Output (GPIO) **369**

Copyright © 2012–2019, Texas Instruments Incorporated

4.1 General-Purpose Input/Output (GPIO)

4.1.1 Introduction

The M3 GPIO module is composed of nine physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, Port G, Port H, Port J). The GPIO module supports up to 66 programmable input/output pins, depending on the peripherals being used.

The module has the following features:

- Up to 66 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- Fast toggle capable of a change every two clock cycles
- Two means of port access: either Advanced High-Performance Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
  - Weak pull-up resistors
  - Open drain enables
  - Digital input enables

On REV0 of this device, PF6_GPIO38 and PF6_GPIO46 are not available for general-purpose usage but are tied to USB0VBUS and USB0ID functionality. So on REV0 of this device, the total number of available GPIOs is 64 instead of 66.

4.1.2 Signal Description

GPIO signals have alternate hardware functions. Table 4-1 lists the GPIO pins and their analog and digital alternate functions. The USB0VBUS and USB0ID analog signals are configured by clearing the corresponding DEN bit in the GPIO Digital Enable (GPIODEN) register and setting the corresponding AMSEL bit in the GPIO Analog Mode Select (GPIOAMSEL) register. The digital 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.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (GPIOAFSEL=0, GPIODEN=0, GPIOPUR=0, and GPIOPCTL=0). A Power-On-Reset (POR) or asserting $\overline{\text{XRS}}$ puts the pins back to their default state.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-1. GPIO Pins and Alternate Functions

		Digital Function (GPIOPCTL PMCx Bit Field Encoding)										
IO	Analog Function	1	2	3	4	5	6	7	8	9	10	11
PA0_GPIO0		U0Rx							I2C1SCL	U1Rx		
PA1_GPIO1		U0Tx							I2C1SDA	U1Tx		
PA2_GPIO2		SSI0Clk		MII_TXD2								
PA3_GPIO3		SSI0Fss		MII_TXD1								
PA4_GPIO4		SSI0Rx		MII_TXD0		CAN0Rx						
PA5_GPIO5		SSI0Tx		MII_RXDV		CAN0Tx						
PA6_GPIO6		I2C1SCL	CCP1	MII_RXCK			CAN0Rx		USB0EPEN			
PA7_GPIO7		I2C1SDA	CCP4	MII_RXER			CAN0Tx	CCP3	USB0PFLT			
PB0_GPIO8		CCP0				U1Rx						
PB1_GPIO9		CCP2			CCP1	U1Tx						
PB2_GPIO10		I2C0SCL			CCP3	CCP0			USB0EPEN			
PB3_GPIO11		I2C0SDA							USB0PFLT			
PB4_GPIO12					U2Rx	CAN0Rx		U1Rx	EPI0S23			
PB5_GPIO13			CCP5	CCP6	CCP0	CAN0Tx	CCP2	U1Tx	EPI0S22			
PB6_GPIO14		CCP1	CCP7				CCP5					
PB7_GPIO15					NMI			MII_RXD1				
PC4_GPIO68		CCP5		MII_TXD3		CCP2	CCP4		EPI0S2	CCP1		
PC5_GPIO69		CCP1				CCP3	USB0EPEN		EPI0S3			
PC6_GPIO70		CCP3				U1Rx	CCP0	USB0PFLT	EPI0S4			
PC7_GPIO71		CCP4			CCP0	U1Tx	USB0PFLT		EPI0S5			
PD0_GPIO16			CAN0Rx		U2Rx	U1Rx	CCP6	MII_RXDV				
PD1_GPIO17			CAN0Tx		U2Tx	U1Tx	CCP7	MII_TXER			CCP2	
PD2_GPIO18		U1Rx	CCP6		CCP5				EPI0S20			
PD3_GPIO19		U1Tx	CCP7		CCP0				EPI0S21			
PD4_GPIO20		CCP0	CCP3		MII_TXD3						EPI0S19	
PD5_GPIO21		CCP2	CCP4		MII_TXD2					U2Rx	EPI0S28	
PD6_GPIO22					MII_TXD1					U2Tx	EPI0S29	
PD7_GPIO23				CCP1	MII_TXD0						EPI0S30	
PE0_GPIO24			SSI1Clk	CCP3					EPI0S8	USB0PFLT		
PE1_GPIO25			SSI1Fss		CCP2	CCP6			EPI0S9			
PE2_GPIO26		CCP4	SSI1Rx			CCP2			EPI0S24			
PE3_GPIO27		CCP1	SSI1Tx			CCP7			EPI0S25			
PE4_GPIO28		CCP3				U2Tx	CCP2	MII_RXD0				
PE5_GPIO29		CCP5										
PE6_GPIO30												
PE7_GPIO31												
PF0_GPIO32		CAN1Rx			MII_RXCK							
PF1_GPIO33		CAN1Tx			MII_RXER						CCP3	
PF2_GPIO34				MII_PHYINTRn						SSI1Clk		

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-1. GPIO Pins and Alternate Functions (continued)

		Digital Function (GPIOPCTL PMCx Bit Field Encoding)										
PF3_GPIO35				MII_MDC						SSI1Fss		
PF4_GPIO36		CCP0		MII_MDIO					EPI0S12	SSI1Rx		
PF5_GPIO37		CCP2		MII_RXD3					EPI0S15	SSI1Tx		
PF6_GPIO38	USB0VBUS	CCP1		MII_RXD2								
PG0_GPIO40		U2Rx		I2C1SCL				USB0EPEN	EPI0S13			
PG1_GPIO41		U2Tx		I2C1SDA					EPI0S14			
PG2_GPIO42	USB0DM			MII_COL								
PG3_GPIO43				MII_CRS								
PG5_GPIO45	USB0DP	CCP5		MII_TXEN								
PG6_GPIO46	USB0ID			MII_TXCK								
PG7_GPIO47				MII_TXER					CCP5	EPI0S31		
PH0_GPIO48		CCP6		MII_PHYRSTn					EPI0S6			
PH1_GPIO49		CCP7							EPI0S7			
PH2_GPIO50									EPI0S1	MII_TXD3		
PH3_GPIO51					USB0EPEN				EPI0S0	MII_TXD2		
PH4_GPIO52					USB0PFLT				EPI0S10	MII_TXD1		SSI1Clk
PH5_GPIO53									EPI0S11	MII_TXD0		SSI1Fss
PH6_GPIO54									EPI0S26	MII_RXDV		SSI1Rx
PH7_GPIO55				MII_RXCK					EPI0S27			SSI1Tx
PJ0_GPIO56				MII_RXER					EPI0S16			I2C1SCL
PJ1_GPIO57									EPI0S17	USB0PFLT		I2C1SDA
PJ2_GPIO58									EPI0S18	CCP0		
PJ3_GPIO59									EPI0S19		CCP6	
PJ4_GPIO60									EPI0S28		CCP4	
PJ5_GPIO61									EPI0S29		CCP2	
PJ6_GPIO62									EPI0S30		CCP1	
PJ7_GPIO63/ XCLKIN											CCP0	

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-2. GPIO Pins and Alternate Mode Functions

	Digital Function (GPIOPCTL PMCx Bit Field Encoding)			
IO	12	13	14	15
PA0_GPIO0				
PA1_GPIO1				SSI1Fss
PA2_GPIO2				
PA3_GPIO3				SSI1Clk
PA4_GPIO4				
PA5_GPIO5				
PA6_GPIO6				
PA7_GPIO7	MII_RXD1			
PB0_GPIO8		SSI2Tx	CAN1Tx	U4Tx
PB1_GPIO9		SSI2Rx		
PB2_GPIO10		SSI2Clk	CAN1Rx	U4Rx
PB3_GPIO11		SSI2Fss	U1Rx	
PB4_GPIO12			CAN1Tx	SSI1Tx
PB5_GPIO13			CAN1Rx	SSI1Rx
PB6_GPIO14	MII_CRS	I2C0SDA	U1Tx	SSI1Clk
PB7_GPIO15		I2C0SCL	U1Rx	SSI1Fss
PC4_GPIO68				
PC5_GPIO69				
PC6_GPIO70				
PC7_GPIO71				
PD0_GPIO16	MII_RXD2	SSI0Tx	CAN1Tx	USB0EPEN
PD1_GPIO17	MII_COL	SSI0Rx	CAN1Rx	USB0PFLT
PD2_GPIO18		SSI0Clk	U1Tx	CAN0Rx
PD3_GPIO19		SSI0Fss	U1Rx	CAN0Tx
PD4_GPIO20			U3Tx	CAN1Tx
PD5_GPIO21			U3Rx	CAN1Rx
PD6_GPIO22			I2C1SDA	U1Tx
PD7_GPIO23			I2C1SCL	U1Rx
PE0_GPIO24		SSI3Tx	CAN0Rx	SSI1Tx
PE1_GPIO25		SSI3Rx	CAN0Tx	SSI1Rx
PE2_GPIO26		SSI3Clk	U2Rx	SSI1Clk
PE3_GPIO27		SSI3Fss	U2Tx	SSI1Fss
PE4_GPIO28		U0Rx		USB0EPEN
PE5_GPIO29	MII_TXER	U0Tx		USB0PFLT
PE6_GPIO30	MII_MDIO	CAN0Rx		
PE7_GPIO31	MII_RXD3	CAN0Tx		
PF0_GPIO32		I2C0SDA	TRACED2	
PF1_GPIO33		I2C0SCL	TRACED3	
PF2_GPIO34			TRACECLK	XCLKOUT_O
PF3_GPIO35		U0Tx	TRACED0	
PF4_GPIO36		U0Rx		
PF5_GPIO37				
PF6_GPIO38				
PG0_GPIO40	MII_RXD2	U4Rx		
PG1_GPIO41	MII_RXD1	U4Tx		
PG2_GPIO42				
PG3_GPIO43	MII_RXDV		TRACED1	
PG5_GPIO45				
PG6_GPIO46				
PG7_GPIO47				
PH0_GPIO48		SSI3Tx		
PH1_GPIO49	MII_RXD0	SSI3Rx		
PH2_GPIO50		SSI3Clk		
PH3_GPIO51		SSI3Fss		
PH4_GPIO52		U3Tx		

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-2. GPIO Pins and Alternate Mode Functions (continued)

	Digital Function (GPIOPCTL PMCx Bit Field Encoding)			
PH5_GPIO53		U3Rx		
PH6_GPIO54	MII_TXEN	SSI0Tx		
PH7_GPIO55	MII_TXCK	SSI0Rx		
PJ0_GPIO56		SSI0Clk		
PJ1_GPIO57	MII_RXDV	SSI0Fss		
PJ2_GPIO58	MII_RXCK	SSI0Clk	U0Tx	
PJ3_GPIO59	MII_MDC	SSI0Fss	U0Rx	
PJ4_GPIO60	MII_COL	SSI1Clk		
PJ5_GPIO61	MII_CRS	SSI1Fss		
PJ6_GPIO62	MII_PHYINTRn	U2Rx		
PJ7_GPIO63/XCLKIN	MII_PHYRSTn	U2Tx		

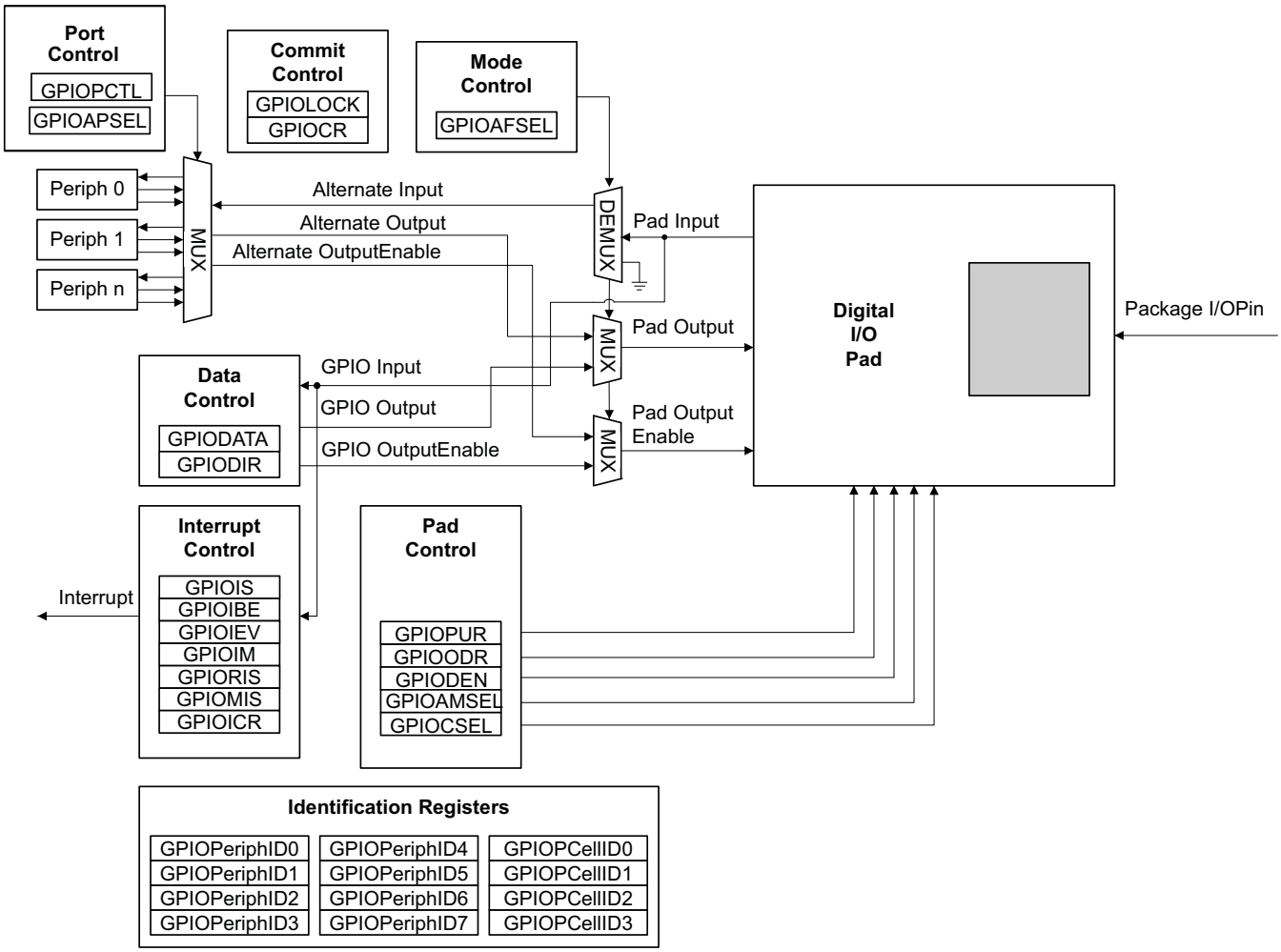
The M3 GPIO Mux also contains an alternate muxing mode. The proper bits in the Alternate Peripheral Select (GPIOAPSEL) register must be set to access these muxing options. The Digital Function (GPIOPCTL) register can then be used to select the mux option.

4.1.3 Functional Description

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 4-1). The microcontroller contains nine ports and therefore nine of these physical GPIO blocks. Note that not all pins may be implemented on every block. Some GPIO pins can function as I/O signals for the on-chip peripheral modules.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-1. Digital I/O Pads



4.1.3.1 Master Control

The M3 GPIO mux is the master, and at boot time, the user needs to allocate which core (M3 or C28) controls which pin. By default, all pins are assigned to the M3 GPIO mux except Group 2 GPIOs. See Section 4.2 for more information.

To select which core controls a GPIO pin, use the GPIO Core Select (GPIOCSEL) register. If the M3 GPIOs are enabled by the GPIODEN register, the M3 can still monitor any GPIO, even if it is mapped to the C28 GPIO mux.

4.1.3.2 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.

4.1.3.2.1 Data Direction Operation

The GPIO Direction (GPIODIR) register 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.

Copyright © 2012–2019, Texas Instruments Incorporated

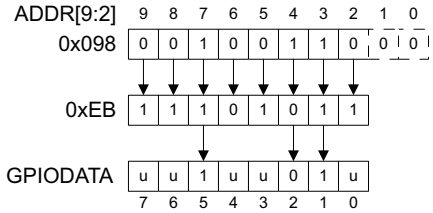
4.1.3.2.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the GPIO Data (GPIODATA) register 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 , where u indicates that data is unchanged by the write.

Figure 4-2. 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 4-3. GPIODATA Read Example

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.3.3 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
- GPIO Interrupt Both Edges (GPIOIBE) register
- GPIO Interrupt Event (GPIOIEV) register

Interrupts are enabled/disabled via the GPIO Interrupt Mask (GPIOIM) register.

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

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

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.

4.1.3.4 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, 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.

4.1.3.5 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7). Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register, GPIO Pull Up Select (GPIOPUR) register, GPIO Core Select (GPIOCSEL) register, and GPIO Digital Enable (GPIODEN) register are not committed to storage unless the GPIO Lock (GPIOLOCK) register has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register have been set.

4.1.3.6 Pad Control

The pad control registers allow software to configure the GPIO pads based on the application requirements. The pad control registers include the GPIOODR, GPIOPUR and GPIODEN registers. These registers control open-drain configuration, pull-up resistors, and digital input enable for each GPIO.

4.1.3.7 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.

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.4 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 existing software. 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.

To use the pins in a particular GPIO port, the clock for the port must be enabled by setting the appropriate GPIO Port bit field (GPIOn) in the RCGC2 register.

On reset, all GPIO pins are configured to be undriven (tri-state): GPIOAFSEL=0, GPIODEN=0 and GPIOPUR=0. Table 4-3 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 4-4 shows how a rising edge interrupt is configured for pin 2 of a GPIO port.

It is recommended to configure the GPIOPCTL register before configuring the GPIOAFSEL register.

For example (referring to the ControlSuite UART examples), the recommended configuration procedure for UART is as follows:

```
GPIOPinConfigure(GPIO_PA0_U0RX);
GPIOPinConfigure(GPIO_PA1_U0TX);
GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
```

rather than:

```
GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
GPIOPinConfigure(GPIO_PA0_U0RX);
GPIOPinConfigure(GPIO_PA1_U0TX);
```

While the latter one would work without any problem on REV0, the behavior changed as described below from REVA onward.

On F28M35x REV0 – if user selects GPIOAFSEL with GPIOPCTL=0, then the respective IO is tri-stated.

On F28M35x REVA/B– if user selects GPIOAFSEL with GPIOPCTL=0, then the respective IO is driven LOW.

Table 4-3. GPIO Pad Configuration Examples

Configuration	GPIO Register Bit Value[1]						
	AFSEL	DIR	ODR	DEN	PUR	CSEL	APSEL
Digital Input (GPIO)	0	0	0	1	?	0	0
Digital Output (GPIO)	0	1	0	1	?	0	0
Open Drain Output (GPIO)	0	1	1	1	X	0	0
Open Drain Input/Output (I2C)	1	X	1	1	X	0	?
Digital Input (Timer CCP)	1	X	0	1	?	0	0
Digital Output (Timer PWM)	1	X	0	1	?	0	0
Digital Input/Output (SSI)	1	X	0	1	?	0	?
Digital Input/Output (UART)	1	X	0	1	?	0	?

(1) X=Ignored (don't care bit); ?=Can be either 0 or 1, depending on the configuration

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-4. GPIO Interrupt Configuration Example

Register	Desired Interrupt Event Trigger	Pin 2 Bit Value[1]							
		7	6	5	4	3	2	1	0
GPIOIS	0=edge1=level	X	X	X	X	X	0	X	X
GPIOIBE	0=single edge1=both edges	X	X	X	X	X	0	X	X
GPIOIEV	0=Low level, or falling edge1=High level, or rising edge	X	X	X	X	X	1	X	X
GPIOIM	0=masked1=not masked	0	0	0	0	0	1	0	0

(1) X=Ignored (don't care bit)

4.1.5 Register Map

Table 4-5 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 existing software. 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.

NOTE: The GPIO registers in this chapter are duplicated in each GPIO block; however, depending on the block, all eight bits may not be connected to a GPIO pad. In those cases, writing to unconnected bits has no effect, and reading unconnected bits returns no meaningful data.

The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A (APB): 0x4000.4000
- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (APB): 0x4000.5000
- GPIO Port B (AHB): 0x4005.9000
- GPIO Port C (APB): 0x4000.6000
- GPIO Port C (AHB): 0x4005.A000
- GPIO Port D (APB): 0x4000.7000
- GPIO Port D (AHB): 0x4005.B000
- GPIO Port E (APB): 0x4002.4000
- GPIO Port E (AHB): 0x4005.C000
- GPIO Port F (APB): 0x4002.5000
- GPIO Port F (AHB): 0x4005.D000
- GPIO Port G (APB): 0x4002.6000
- GPIO Port G (AHB): 0x4005.E000
- GPIO Port H (APB): 0x4002.7000
- GPIO Port H (AHB): 0x4005.F000
- GPIO Port J (APB): 0x4003.D000
- GPIO Port J (AHB): 0x4006.0000
- GPIO Port K (AHB): 0x4006.1000

Copyright © 2012–2019, Texas Instruments Incorporated

Important: All GPIO pins are configured as GPIOs and tri-stated by default (GPIOAFSEL=0, GPIODEN=0, GPIOPUR=0, and GPIOPCTL=0. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{XRS}$ puts the pins back to their default state.

NOTE: The default register type for the GPIOCR register is RO for all GPIO pins with the exception of the NMI pin. The PB7 pin is currently the only GPIO that is protected by the GPIOCR register. Because of this, the register type for GPIO Port B7 is R/W.

The default reset value for the GPIOCR register is 0x0000.00FF for all GPIO pins, with the exception of the NMI pin. To ensure that the NMI pin is not accidentally programmed as the non-maskable interrupt pin, it defaults to non-committable. Because of this, the default reset value of GPIOCR for GPIO Port B is 0x0000.007F.

Table 4-5. GPIO Register Map

Offset	Name	Type	Reset	Description
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data
0X400	GPIODIR	R/W	0x0000.0000	GPIO Direction
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select
0x510	GPIOPUR	R/W	-	GPIO Pull-Up Select
0x51C	GPIODEN	R/W	-	GPIO Digital Enable
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock
0x524	GPIOCR	-	-	GPIO Commit
0x528	GPIOAMSEL	R/W	0x0000.0000	GPIO Analog Mode Select
0x52C	GPIOPCTL	R/W	-	GPIO Port Control
0x530	GPIOAPSEL	R/W	0x0000.0000	GPIO Alternate Peripheral Select
0x534	GPIOCSEL	R/W	0x0000.0000	GPIO Core Select
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-5. GPIO Register Map (continued)

0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3

4.1.6 Register Descriptions

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

4.1.6.1 GPIO Data (GPIODATA) Register, 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 .

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.

Figure 4-4. GPIO Data (GPIODATA) Register

31	16
Reserved	
R-0	

15	8	7	0
Reserved		DATA	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-6. GPIO Data (GPIODATA) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	DATA	00h	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 Section 4.1.3.2.2 for examples of reads and writes.

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.2 GPIO Direction (GPIODIR) Register, 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.

Figure 4-5. GPIO Direction (GPIODIR) Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DIR							
R-0								R/W-0							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-7. GPIO Direction (GPIODIR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	DIR		GPIO Data Direction
		0	Corresponding pin as an input.
		1	Corresponding pin as an output.

4.1.6.3 GPIO Interrupt Sense (GPIOIS) Register, 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.

Figure 4-6. GPIO Interrupt Sense (GPIOIS) Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IS							
R-0								R/W-0							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-8. GPIO Interrupt Sense (GPIOIS) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	IS		GPIO Interrupt Sense
		0	The edge on the corresponding pin is detected (edge-sensitive).
		1	The level on the corresponding pin is detected (level-sensitive).

Copyright © 2012–2019, Texas Instruments Incorporated

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

The GPIOIBE register allows both edges to cause interrupts. When the corresponding bit in the GPIO Interrupt Sense (GPIOIS) register 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. Clearing a bit configures the pin to be controlled by the GPIOIEV register. All bits are cleared by a reset.

Figure 4-7. GPIO Interrupt Both Edges (GPIOIBE) Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IBE							
R-0								R/W-0							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-9. GPIO Interrupt Both Edges (GPIOIBE) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	IBE		GPIO Interrupt Both Edges
		0	Interrupt generation is controlled by the GPIO Interrupt Event (GPIOIEV) register.
		1	Both edges on the corresponding pin trigger an interrupt.

4.1.6.5 GPIO Interrupt Event (GPIOIEV) Register, 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. 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.

Figure 4-8. GPIO Interrupt Event (GPIOIEV) Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IEV							
R-0								R/W-0							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-10. GPIO Interrupt Event (GPIOIEV) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	IEV		GPIO Interrupt Event
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.6 GPIO Interrupt Mask (GPIOIM) Register, 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.

Figure 4-9. GPIO Interrupt Mask (GPIOIM) Register

31	16
Reserved	
R-0	

15	8	7	0
Reserved		IME	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-11. GPIO Interrupt Mask (GPIOIM) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	IME		GPIO Interrupt Mask Enable
		0	The interrupt from the corresponding pin is masked.
		1	The interrupt from the corresponding pin is sent to the interrupt controller.

4.1.6.7 GPIO Raw Interrupt Status (GPIORIS) Register, 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 is set, the interrupt is sent to the interrupt controller. Bits read as zero indicate that corresponding input pins have not initiated an interrupt. A bit in this register can be cleared by writing a 1 to the corresponding bit in the GPIO Interrupt Clear (GPIOICR) register.

Figure 4-10. GPIO Raw Interrupt Status (GPIORIS) Register

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RIS	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-12. GPIO Raw Interrupt Status (GPIORIS) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	RIS		GPIO Interrupt Raw Status
		0	An interrupt condition has not occurred on the corresponding pin.
		1	An interrupt condition has occurred on the corresponding pin.
			A bit is cleared by writing a 1 to the corresponding bit in the GPIOICR register.

4.1.6.8 GPIO Masked Interrupt Status (GPIOMIS) Register, 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.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-11. GPIO Masked Interrupt Status (GPIOMIS) Register

31	16
Reserved	
R-0	

15	8	7	0
Reserved		MIS	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-13. GPIO Masked Interrupt Status (GPIOMIS) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	MIS		GPIO Masked Interrupt Status
		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.
			A bit is cleared by writing a 1 to the corresponding bit in the GPIOICR register.

4.1.6.9 GPIO Interrupt Clear (GPIOICR) Register, offset 0x41C

The GPIOICR register is the interrupt clear register. Writing a 1 to a bit in this register clears the corresponding interrupt bit in the GPIORIS and GPIOMIS registers. Writing a 0 has no effect.

Figure 4-12. GPIO Interrupt Clear (GPIOICR) Register

31	16
Reserved	
R-0	

15	8	7	0
Reserved		IC	
R-0		W/1C-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-14. GPIO Interrupt Clear (GPIOICR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	IC		GPIO Interrupt Raw Status
		0	The corresponding interrupt is unaffected.
		1	The corresponding interrupt is cleared.

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.10 GPIO Alternate Function Select (GPIOAFSEL) Register, 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.

NOTE: All GPIO pins are configured as GPIOs and tri-stated by default (GPIOAFSEL=0, GPIODEN=0, GPIOPUR=0, and GPIOPCTL=0. A Power-On-Reset $\overline{\text{POR}}$ or asserting $\overline{\text{XRS}}$ puts the pins back to their default state.

CAUTION

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

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7). Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register, GPIO Pull Up Select (GPIOPUR) register, GPIO Core Select (GPIOCSEL) register, and GPIO Digital Enable (GPIODEN) register are not committed to storage unless the GPIO Lock (GPIOLOCK) register has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register have been set.

When using the I2C module, in addition to setting the GPIOAFSEL register bits for the I2C clock and data pins, the pins should be set to open drain using the GPIO Open Drain Select (GPIOODR) register (see examples in Section 4.1.4).

Figure 4-13. GPIO Alternate Function Select (GPIOAFSEL) Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								AFSEL							
R-0								R/W-0							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-15. GPIO Alternate Function Select (GPIOAFSEL) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	AFSEL		GPIO Alternate Function Select
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.11 GPIO Open Drain Select (GPIOODR) Register, 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. The GPIO acts as an open-drain input if the corresponding bit in the GPIODIR register is cleared. If open drain is selected while the GPIO is configured as an input, the GPIO will remain an input and the open-drain selection has no effect until the GPIO is changed to an output.

When using the I2C module, in addition to configuring the pin to open drain, the GPIO Alternate Function Select (GPIOAFSEL) register bits for the I2C clock and data pins should be set (see examples in Section 4.1.4).

Figure 4-14. GPIO Open Drain Select (GPIOODR) Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ODE							
R-0								R/W-0							

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-16. GPIO Open Drain Select (GPIOODR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	ODE		Output Pad Open Drain Enable
		0	The corresponding pin is not configured as open drain.
		1	The corresponding pin is configured as open drain.

4.1.6.12 GPIO Pull-Up Select (GPIOPUR) Register, 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. Write access to this register is protected with the GPIOCR register. Bits in GPIOCR that are cleared prevent writes to the equivalent bit in this register.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (GPIOAFSEL=0, GPIODEN=0, GPIOPUR=0, and GPIOPCTL=0. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{XRS}}$ puts the pins back to their default state.

NOTE: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7). Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register, GPIO Pull Up Select (GPIOPUR) register, GPIO Select Core (GPIOCSEL) register, and GPIO Digital Enable (GPIODEN) register are not committed to storage unless the GPIO Lock (GPIOLOCK) register has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register have been set.

Figure 4-15. GPIO Pull-Up Select (GPIOPUR) Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PUE							
R-0								R/W-0							

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-17. GPIO Pull-Up Select (GPIOPUR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PUE		Pad Weak Pull-Up Enable
		0	The corresponding pin is not affected.
		1	The corresponding pin has a weak pull-up resistor.
			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.

4.1.6.13 GPIO Digital Enable (GPIODEN) Register, offset 0x51C

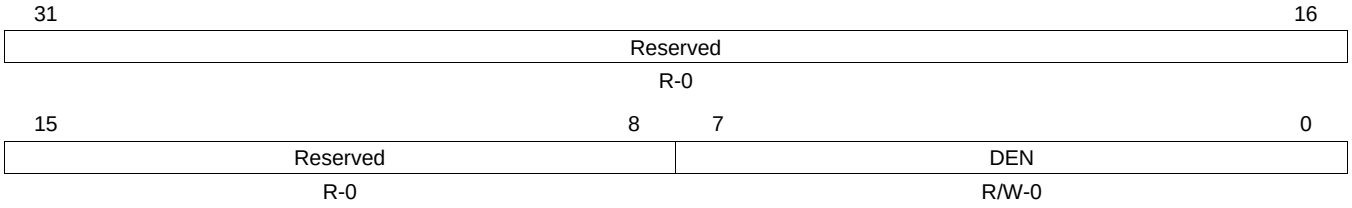
NOTE: Pins configured as digital inputs are Schmitt-triggered.

The GPIODEN register is the digital enable register. By default, all GPIO signals are configured out of reset to be undriven (tristate). Their digital function is disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use the pin as a digital input or output (either GPIO or alternate function), the corresponding GPIODEN bit must be set.

Important All GPIO pins are configured as GPIOs and tri-stated by default (GPIOAFSEL=0, GPIODEN=0, GPIOPUR=0, and GPIOPCTL=0. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{XRS}}$ puts the pins back to their default state.

NOTE: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7). Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register, GPIO Pull Up Select (GPIOPUR) register, GPIO Core Select (GPIOCSEL) register, and GPIO Digital Enable (GPIODEN) register are not committed to storage unless the GPIO Lock (GPIOLOCK) register has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register have been set.

Figure 4-16. GPIO Digital Enable (GPIODEN) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-18. GPIO Digital Enable (GPIODEN) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	DEN		Pad Weak Pull-Up Enable
		0	The digital functions for the corresponding pin are disabled.
		1	The digital functions for the corresponding pin are enabled.

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.14 GPIO Lock (GPIOLOCK) Register, offset 0x520

The GPIOLOCK register enables write access to the GPIOCR register. 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.

Figure 4-17. GPIO Lock (GPIOLOCK) Register

31															16
LOCK															
R/W-0															
15															0
LOCK															
R/W-0															

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-19. GPIO Lock (GPIOLOCK) Register Field Descriptions

Bit	Field	Value	Description
31-0	LOCK		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:
		0	The GPIOCR register is unlocked and may be modified.
		1	The GPIOCR register is locked and may not be modified.

4.1.6.15 GPIO Commit (GPIOCR) Register, offset 0x524

The GPIOCR register is the commit register. The value of the GPIOCR register determines which bits of the GPIOAFSEL, GPIOPUR, GPIOCSEL, 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, GPIOCSEL, 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, GPIOCSEL, 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 hardware. By initializing the bit of the GPIOCR register to 0 for PB7, the NMI can only be converted to a GPIO through a deliberate set of writes to the GPIOLOCK, GPIOCR, and the corresponding registers.

Because this protection is currently only implemented on the NMI pin on PB7, 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, GPIOCSEL, or GPIODEN register bits of these other pins.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-18. GPIO Commit (GPIOCR) Register

31								16
Reserved								
R-0								
15				8	7			0
Reserved					CR			
R-0					-			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-20. GPIO Commit (GPIOCR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CR		GPIO Commit
		0	The corresponding GPIOAFSEL, GPIOPUR, GPIOCSEL, or GPIODEN bits cannot be written.
		1	The corresponding GPIOAFSEL, GPIOPUR, GPIOCSEL, 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, PB7. This pin is currently the only GPIO that is protected by the GPIOCR register. Because of this, the register type for GPIO Port B7 is R/W.
			The default reset value for the GPIOCR register is 0x0000.00FF for all GPIO pins, with the exception of the NMI pin, PB7. To ensure that the NMI pin is not accidentally programmed as the non-maskable interrupt pin, it defaults to non-committable. Because of this, the default reset value of GPIOCR for GPIO Port B is 0x0000.007F.

4.1.6.16 GPIO Analog Mode Select (GPIOAMSEL) Register, offset 0x528

The GPIOAMSEL register selects the analog function of a pin. The appropriate bits need to be set to enable the USB0VBUS and USB0ID signals for USB0 to function correctly.

Figure 4-19. GPIO Analog Mode Select (GPIOAMSEL) Register

31								16
Reserved								
R-0								
15				8	7	4	3	0
Reserved					GPIOAMSEL		Reserved	
R-0					R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-21. GPIO Analog Mode Select (GPIOAMSEL) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-4	GPIOAMSEL		GPIO Analog Mode Select
		0	The analog function of the pin is disabled 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.
3-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.17 GPIO Port Control (GPIOPCTL) Register, 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 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.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (GPIOAFSEL=0, GPIODEN=0, GPIOPUR=0, and GPIOPCTL=0. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{XRS}}$ puts the pins back to their default state.

Figure 4-20. GPIO Port Control (GPIOPCTL) Register

31	28	27	24	23	20	19	16
PMC7		PMC6		PMC5		PMC4	
R/W		R/W		R/W		R/W	

15	12	11	8	7	4	3	0
PMC3		PMC2		PMC1		PMC0	
R/W		R/W		R/W		R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-22. GPIO Port Control (GPIOPCTL) Register Field Descriptions

Bit	Field	Value	Description
31-28	PMC7		Port Mux Control 7 This field controls the configuration for GPIO pin 7.
27-24	PMC6		Port Mux Control 6 This field controls the configuration for GPIO pin 6.
23-20	PMC5		Port Mux Control 5 This field controls the configuration for GPIO pin 5.
19-16	PMC4		Port Mux Control 4 This field controls the configuration for GPIO pin 4.
15-12	PMC3		Port Mux Control 3 This field controls the configuration for GPIO pin 3.
11-8	PMC2		Port Mux Control 2 This field controls the configuration for GPIO pin 2.
7-4	PMC1		Port Mux Control 1 This field controls the configuration for GPIO pin 1.
3-0	PMC0		Port Mux Control 0 This field controls the configuration for GPIO pin 0.

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.18 GPIO Alternate Peripheral Select (GPIOAPSEL) Register, offset 0x530

The GPIOAPSEL register is used to access the M3 GPIO alternate peripheral muxing options. When these bits are set, values of 0x0 - 0xF are valid values in the GPIOPCTL PMCx bit fields. See Section 4.2.6 for alternate muxing options.

Figure 4-21. GPIO Alternate Peripheral Select (GPIOAPSEL) Register

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

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
APSEL7	APSEL6	APSEL5	APSEL4	APSEL3	APSEL2	APSEL1	APSEL0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-23. GPIO Alternate Peripheral Select (GPIOAPSEL) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7	APSEL7		Alternate peripheral select 7
		0	Alternate peripheral mode disabled
		1	Alternate peripheral mode enable
6	APSEL6		Alternate peripheral select 6
		0	Alternate peripheral mode disabled
		1	Alternate peripheral mode enable
5	APSEL5		Alternate peripheral select 5
		0	Alternate peripheral mode disabled
		1	Alternate peripheral mode enable
4	APSEL4		Alternate peripheral select 4
		0	Alternate peripheral mode disabled
		1	Alternate peripheral mode enable
3	APSEL3		Alternate peripheral select 3
		0	Alternate peripheral mode disabled
		1	Alternate peripheral mode enable
2	APSEL2		Alternate peripheral select 2
		0	Alternate peripheral mode disabled
		1	Alternate peripheral mode enable
1	APSEL1		Alternate peripheral select 1
		0	Alternate peripheral mode disabled
		1	Alternate peripheral mode enable
0	APSEL0		Alternate peripheral select 0
			Alternate peripheral mode disabled
			Alternate peripheral mode enable

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.19 GPIO Core Select (GPIOCSEL) Register, offset 0x534

The GPIOCSEL register selects which core controls the pin. If the M3 GPIOs are enabled by the GPIODEN register, the M3 can still monitor any GPIO, even if it is mapped to the C28 GPIO mux.

NOTE: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7). Writes to protected bits of the GPIO Alternate Function Select (GPIOAFSEL) register, GPIO Pull Up Select (GPIOPUR) register, GPIO Core Select (GPIOCSEL) register, and GPIO Digital Enable (GPIODEN) register are not committed to storage unless the GPIO Lock (GPIOLOCK) register has been unlocked and the appropriate bits of the GPIO Commit (GPIOCR) register have been set.

Figure 4-22. GPIO Core Select (GPIOCSEL) Register

31								16
Reserved								
R-0								

15	8	7	6	5	4	3	2	1	0
Reserved		CSEL7	CSEL6	CSEL5	CSEL4	CSEL3	CSEL2	CSEL1	CSEL0
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after rese

Table 4-24. GPIO Core Select (GPIOCSEL) Register Field Descriptions

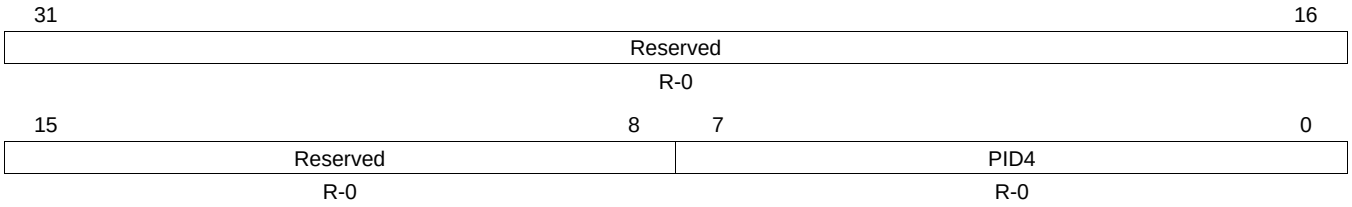
Bit	Field	Value	Description
31-8	Reserved		Reserved
7	CSEL7		Core select 7
		0	Selects M3 GPIO mux
		1	Selects C28 GPIO mux
6	CSEL6		Core select 6
		0	Selects M3 GPIO mux
		1	Selects C28 GPIO mux
5	CSEL5		Core select 5
		0	Selects M3 GPIO mux
		1	Selects C28 GPIO mux
4	CSEL4		Core select 4
		0	Selects M3 GPIO mux
		1	Selects C28 GPIO mux
3	CSEL3		Core select 3
		0	Selects M3 GPIO mux
		1	Selects C28 GPIO mux
2	CSEL2		Core select 2
		0	Selects M3 GPIO mux
		1	Selects C28 GPIO mux
1	CSEL1		Core select 1
		0	Selects M3 GPIO mux
		1	Selects C28 GPIO mux
0	CSEL0		Core select 0
		0	Selects M3 GPIO mux
		0	Selects C28 GPIO mux

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.20 GPIO Peripheral Identification 4 (GPIOPeriphID4) Register, 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.

Figure 4-23. GPIO Peripheral Identification 4 (GPIOPeriphID4) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-25. GPIO Peripheral Identification 4 (GPIOPeriphID4) Register Field Descriptions

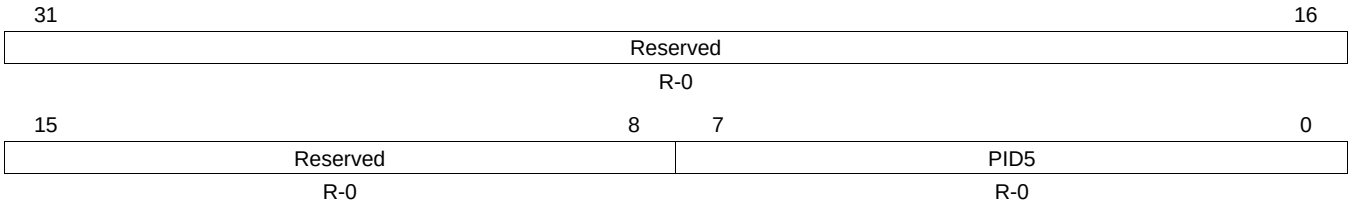
Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID4		GPIO Peripheral ID Register [7:0]

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.21 GPIO Peripheral Identification 5 (GPIOPeriphID5) Register, 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.

Figure 4-24. GPIO Peripheral Identification 5 (GPIOPeriphID5) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-26. GPIO Peripheral Identification 5 (GPIOPeriphID5) Register Field Descriptions

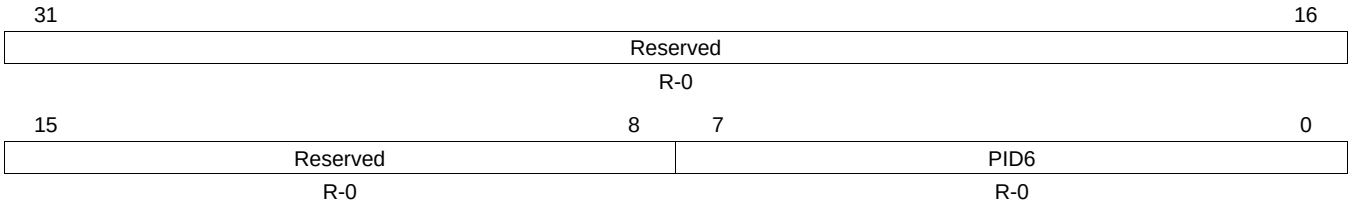
Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID5		GPIO Peripheral ID Register [15:8]

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.22 GPIO Peripheral Identification 6 (GPIOPeriphID6) Register, 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.

Figure 4-25. GPIO Peripheral Identification 6 (GPIOPeriphID6) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-27. GPIO Peripheral Identification 6 (GPIOPeriphID6) Register Field Descriptions

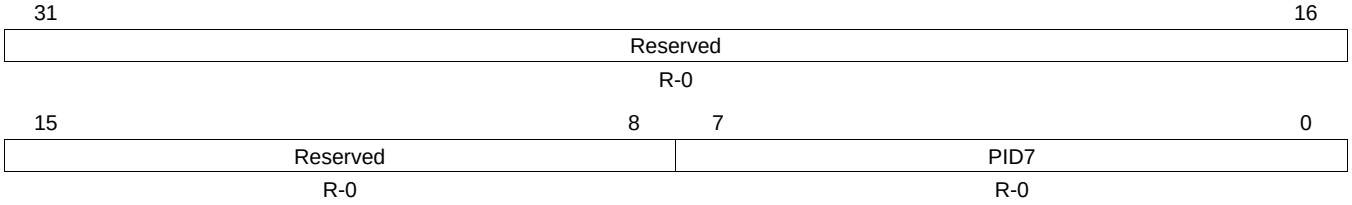
Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID6		GPIO Peripheral ID Register [23:16]

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.23 GPIO Peripheral Identification 7 (GPIOPeriphID7) Register, 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.

Figure 4-26. GPIO Peripheral Identification 7 (GPIOPeriphID7) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-28. GPIO Peripheral Identification 7 (GPIOPeriphID7) Register Field Descriptions

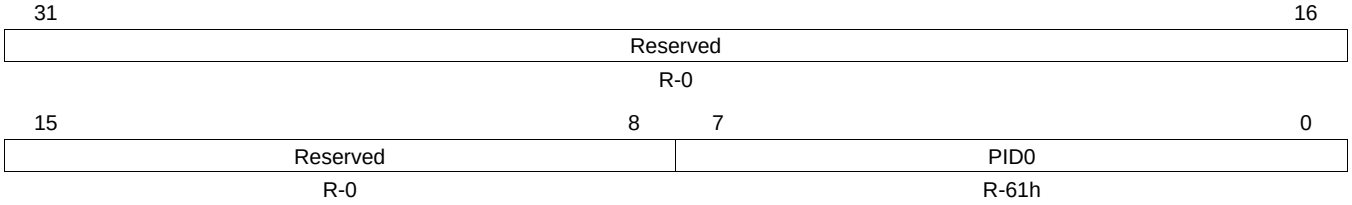
Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID7		GPIO Peripheral ID Register [31:24]

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.24 GPIO Peripheral Identification 0 (GPIOPeriphID0) Register, 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.

Figure 4-27. GPIO Peripheral Identification 0 (GPIOPeriphID0) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

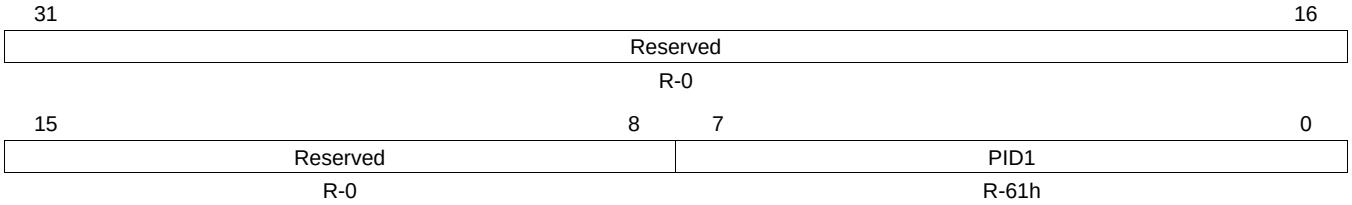
Table 4-29. GPIO Peripheral Identification 0 (GPIOPeriphID0) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID0		GPIO Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

4.1.6.25 GPIO Peripheral Identification 1 (GPIOPeriphID1) Register, 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.

Figure 4-28. GPIO Peripheral Identification 1 (GPIOPeriphID1) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-30. GPIO Peripheral Identification 1 (GPIOPeriphID1) Register Field Descriptions

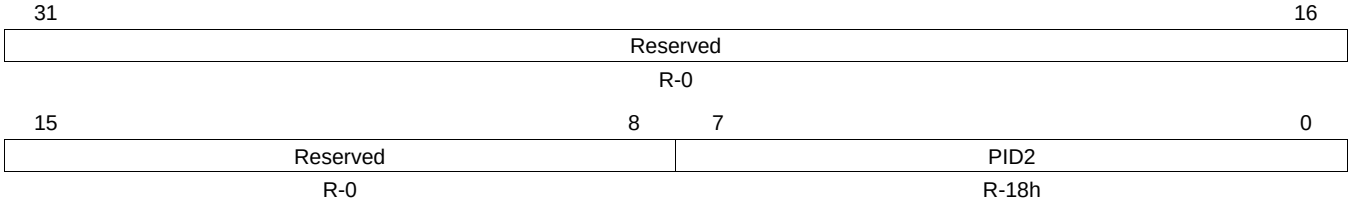
Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID1		GPIO Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Copyright © 2012–2019, Texas Instruments Incorporated

4.1.6.26 GPIO Peripheral Identification 2 (GPIOPeriphID2) Register, 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.

Figure 4-29. GPIO Peripheral Identification 2 (GPIOPeriphID2) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

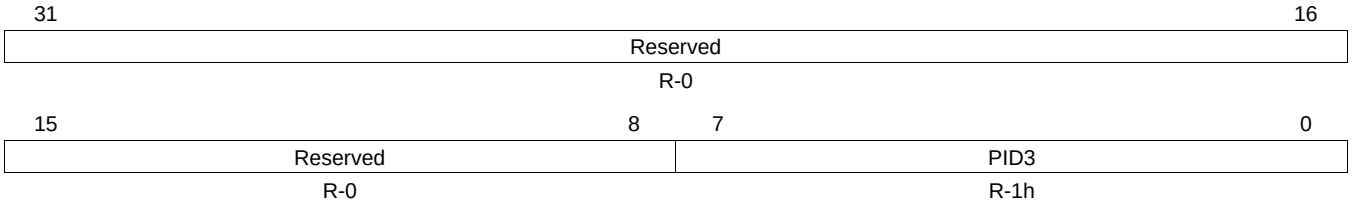
Table 4-31. GPIO Peripheral Identification 2 (GPIOPeriphID2) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID2		GPIO Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

4.1.6.27 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.

Figure 4-30. GPIO Peripheral Identification 3 (GPIOPeriphID3) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-32. GPIO Peripheral Identification 3 (GPIOPeriphID3) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID3		GPIO Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

4.1.6.28 GPIO PrimeCell Identification 0 (GPIOPCellID0) Register, 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.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-31. GPIO PrimeCell Identification 0 (GPIOPCellID0) Register

31	16
Reserved	
R-0	

15	8	7	0
Reserved		CID0	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-33. GPIO PrimeCell Identification 0 (GPIOPCellID0) Register Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID0		GPIO PrimeCell ID Register [7:0]. Provides software a standard cross-peripheral identification system.

4.1.6.29 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.

Figure 4-32. GPIO PrimeCell Identification 1 (GPIOPCellID1) Register

31	16
Reserved	
R-0	

15	8	7	0
Reserved		CID1	
R-0		R-0xF0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-34. GPIO PrimeCell Identification 1 (GPIOPCellID1) Register Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID1		GPIO PrimeCell ID Register [15:8]. Provides software a standard cross-peripheral identification system.

4.1.6.30 GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8

register type RO, reset 0x0000.0005

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.

Figure 4-33. GPIO PrimeCell Identification 2 (GPIOPCellID2) Register

31	16
Reserved	
R-0	

15	8	7	0
Reserved		CID2	
R-0		R-0x5h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-35. GPIO PrimeCell Identification 2 (GPIOPCellID2) Register Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID2		GPIO PrimeCell ID Register [23:16]. Provides software a standard cross-peripheral identification system.

4.1.6.31 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.

Figure 4-34. GPIO PrimeCell Identification 3 (GPIOPCellID3) Register

31	16
Reserved	
R-0	

15	8	7	0
Reserved		CID3	
R-0		R-0xB1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-36. GPIO PrimeCell Identification 3 (GPIOPCellID3) Register Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID3		GPIO PrimeCell ID Register [31:24]. Provides software a standard cross-peripheral identification system.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2 C28 General-Purpose Input/Output (GPIO)

4.2.1 Introduction

The GPIO multiplexing (MUX) registers are used to select the operation of shared pins. The pins are named by their general purpose I/O name (i.e., GPIO0). These pins can be individually selected to operate as digital I/O, referred to as GPIO, or connected to one of up to three peripheral I/O signals (via the GPxMUXn registers). If selected for digital I/O mode, registers are provided to configure the pin direction (via the GPxDIR registers). You can also qualify the input signals to remove unwanted noise (via the GPxQSELn and GPxCTRL registers).

4.2.2 GPIO Module Overview

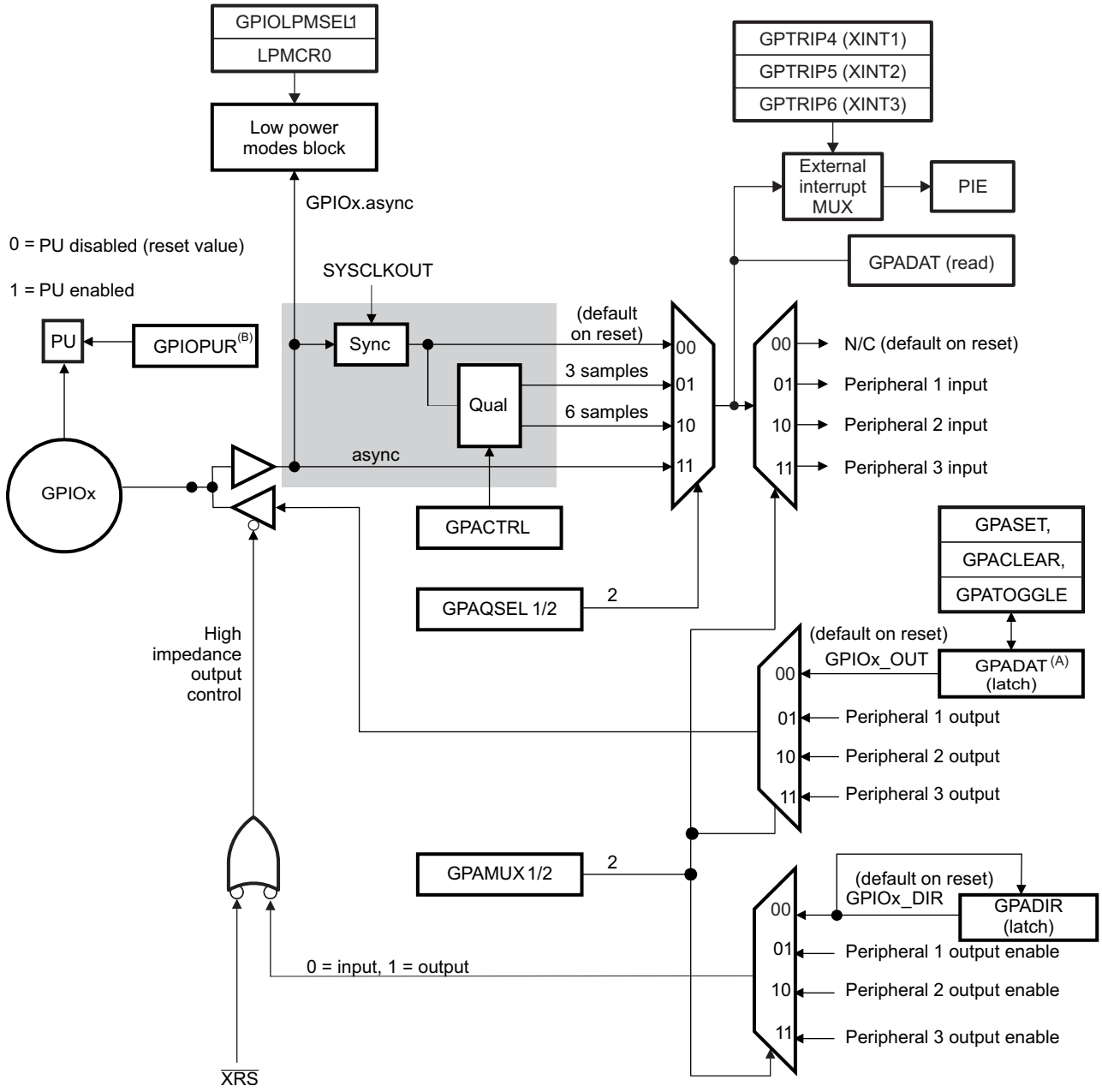
Up to three independent peripheral signals are multiplexed on a single GPIO-enabled pin in addition to individual pin bit-I/O capability. There are six I/O ports:

- Port A consists of GPIO0-GPIO31
- Port B consists of GPIO32-GPIO63
- Port C consists of GPIO68-GPIO71
- Port E consists of GPIO128-GPIO135
- Analog Port 1 consists of AI00-AI015
- Analog Port 2 consists of AIO16-AIO31

Figure 4-35 through Figure 4-38 shows the basic modes of operation for the GPIO module.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-35. GPIO0 to GPIO31 Multiplexing Diagram



A GPxDAT latch/read are accessed at the same memory location.

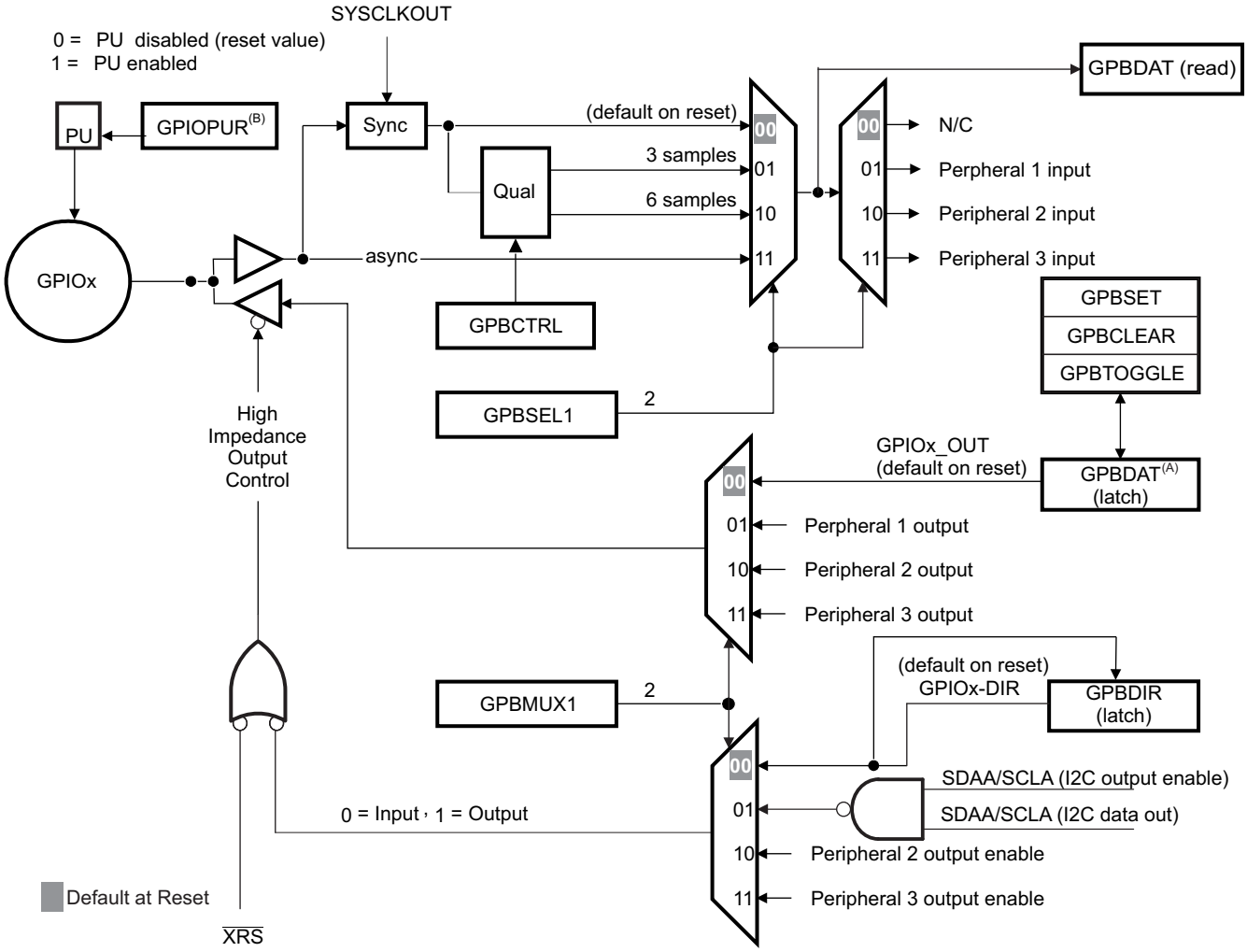
B Pull-up selection is only controlled by the M3 GPIO registers except GPIO128-GPIO135, which is controlled by the GPEPUD register.

Notes:

- Note the bit polarity difference between GPIOPUR and GPEPUD registers when enabling pullups.
- Open drain selection is only controlled by the M3 GPIO registers.
- The appropriate bits in the GPIOCSEL registers (M3 GPIO registers) must be set to use the C28 GPIOs. If the GPIO is set as an M3 GPIO, the C28 GPIO MUX inputs are still active and can be read.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-36. GPIO32, GPIO33 Multiplexing Diagram



A GPxDAT latch/read are accessed at the same memory location.

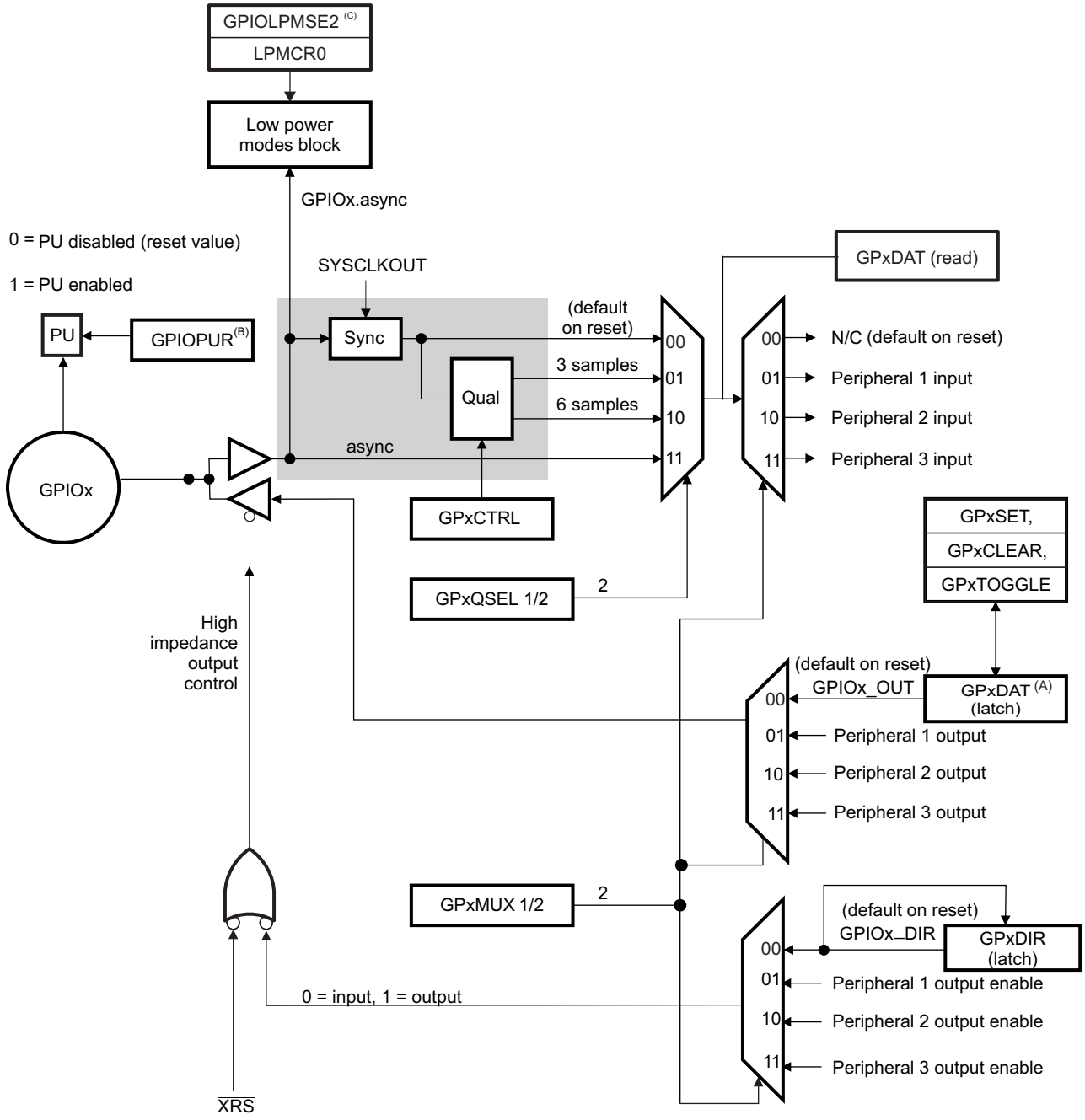
B Pull-up selection is only controlled by the M3 GPIO registers except GPIO128-GPIO135, which is controlled by the GPEPUD register.

Notes:

- Note the bit polarity difference between GPIOPUR and GPEPUD registers when enabling pullups.
- Open drain selection is only controlled by the M3 GPIO registers
- The appropriate bits in the GPIOCSEL registers (M3 GPIO registers) must be set to use the C28 GPIOs. If the GPIO is set as an M3 GPIO, the C28 GPIO MUX inputs are still active and can be read.
- The input qualification circuit is not reset when modes are changed (such as changing from output to input mode). Any state will get flushed by the circuit eventually.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-37. GPIO34, GPIO135 Multiplexing Diagram



A GPxDAT latch/read are accessed at the same memory location.

B Pull-up selection is only controlled by the M3 GPIO registers except GPIO128-GPIO135, which is controlled by the GPEPUD register.

C GPIO32 - GPIO63 only.

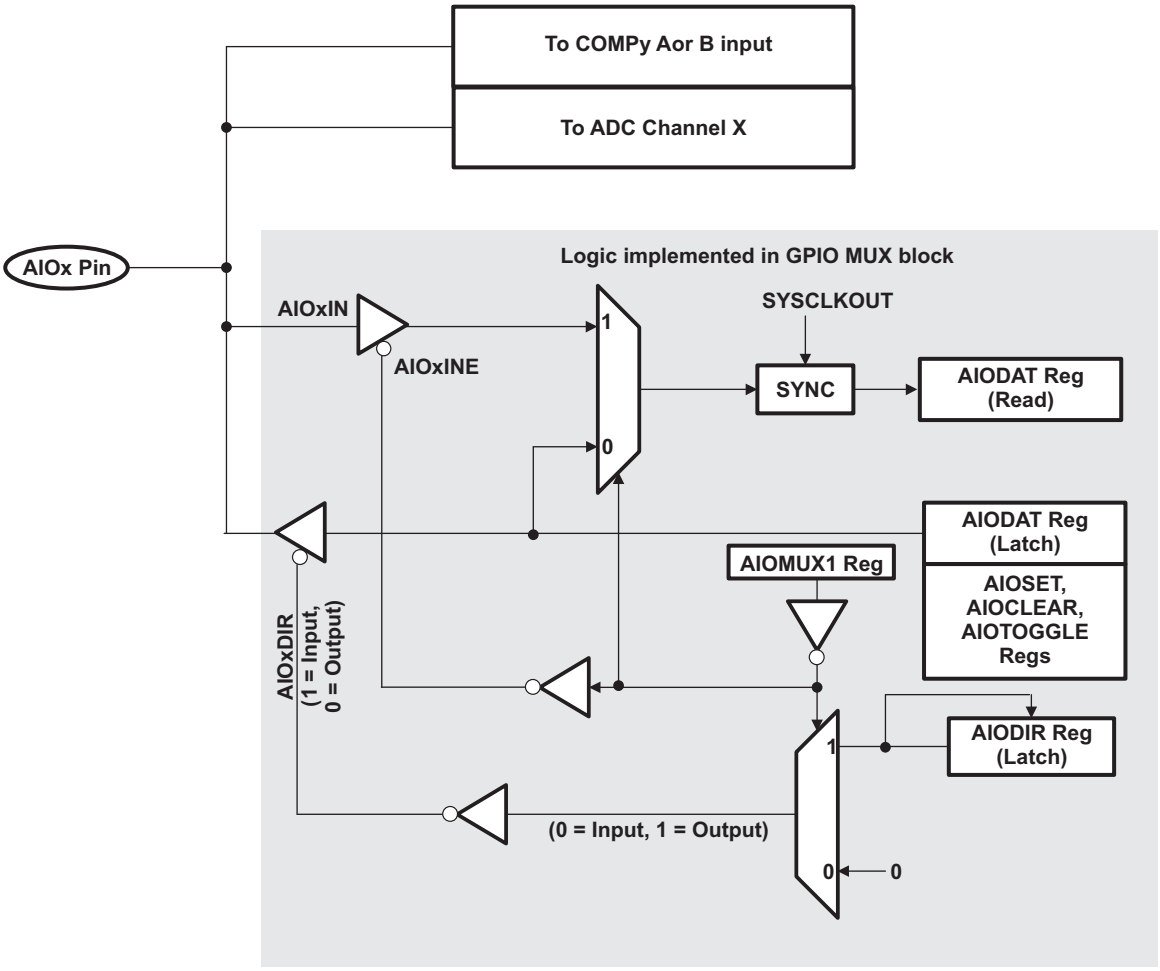
Notes:

- Note the bit polarity difference between GPIOPUR and GPEPUD registers when enabling pullups.
- Open drain selection is only controlled by the M3 GPIO registers
- The appropriate bits in the GPIOCSEL registers (M3 GPIO registers) must be set to use the C28 GPIOs. If the GPIO is set as an M3 GPIO, the C28 GPIO MUX inputs are still active and can be read.
- The input qualification circuit is not reset when modes are changed (such as changing from output to

Copyright © 2012–2019, Texas Instruments Incorporated

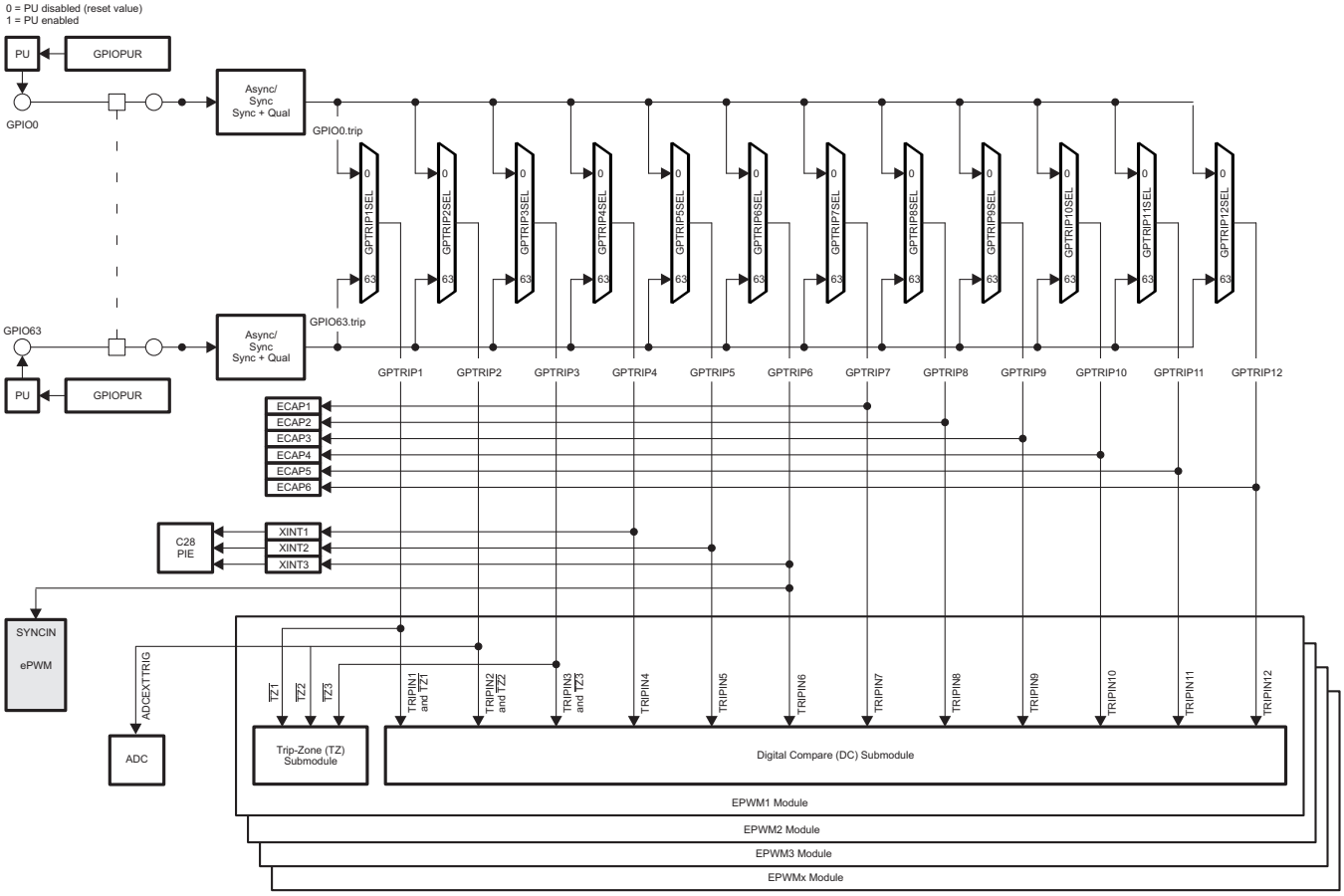
input mode). Any state will get flushed by the circuit eventually.

Figure 4-38. Analog/GPIO Multiplexing



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-39. GPIO MUX-to-Trip Input Connectivity



Any of the 64 GPIO pins can be flexibly mapped to be the trip-zone input or trip inputs to the EPWM trip-zone sub-module and EPWM digital compare sub-module. Refer to the *EPWM* chapter for more information. Any of the 64 GPIO pins can also be mapped to three external interrupts. The GPIO Trip Input Select (GPTRIPxSEL) register defines which GPIO pins get assigned the functionality described above. Refer to the *GPIO* chapter for more information. The GPTRIPxSEL register must also be used to allow ECAP modules to capture data on a pin. Refer to the *ECAP* chapter for more information.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.3 Configuration Overview

The pin function assignments, input qualification, and the external interrupt sources are all controlled by the GPIO configuration control registers. In addition, you can assign pins to wake the device from the HALT and STANDBY low power modes and enable/disable internal pullup resistors. Table 4-37 and Table 4-38 list the registers that are used to configure the GPIO pins to match the system requirements.

Table 4-37. GPIO Control Registers

Name[1]	Address	Size (x16)	Register Description
GPACTRL	0x5F80	2	GPIO A Control Register (GPIO0 - GPIO31)
GPAQSEL1	0x5F82	2	GPIO A Qualifier Select 1 Register (GPIO0 - GPIO15)
GPAQSEL2	0x5F84	2	GPIO A Qualifier Select 2 Register (GPIO16 - GPIO31)
GPAMUX1	0x5F86	2	GPIO A MUX 1 Register (GPIO0 - GPIO15)
GPAMUX2	0x5F88	2	GPIO A MUX 2 Register (GPIO16 - GPIO31)
GPADIR	0x5F8A	2	GPIO A Direction Register (GPIO0 - GPIO31)
GPBCTRL	0x5F90	2	GPIO B Control Register (GPIO32 - GPIO63)
GPBQSEL1	0x5F92	2	GPIO B Qualifier Select 1 Register (GPIO32 - GPIO47)
GPBQSEL2	0x5F94	2	GPIO B Qualifier Select 2 Register (GPIO48 - GPIO63)
GPBMUX1	0x5F96	2	GPIO B MUX 1 Register (GPIO32 - GPIO47)
GPBMUX2	0x5F98	2	GPIO B MUX 2 Register (GPIO48 - GPIO63)
GPBDIR	0x5F9A	2	GPIO B Direction Register (GPIO32 - GPIO63)
GPCCTRL	0x5FA0	2	GPIO C Control Register (GPIO68 to GPIO71)
GPCQSEL1	0x5FA2	2	GPIO C Qualifier Select 1 Register (GPIO68 - GPIO71)
GPCMUX1	0x5FA6	2	GPIO C MUX 1 Register (GPIO68 - GPIO71)
GPCDIR	0x5FAA	2	GPIO C Direction Register (GPIO68 - GPIO71)
GPECTRL	0x6F80	2	GPIO E Control Register (GPIO128 - GPIO135)
GPEQSEL1	0x6F82	2	GPIO E Qualifier Select 1 Register (GPIO128 - GPIO135)
GPEMUX1	0x6F86	2	GPIO E MUX 1 Register (GPIO128 - GPIO135)
GPEDIR	0x6F8A	2	GPIO E Direction Register (GPIO128 - GPIO135)
GPEPUD	0x6F8C	2	GPIO E Pull Up Disable Register (GPIO128 - GPIO135)
AIOMUX1	0x6FB6	2	Analog IO MUX 1 Register (AIO0 to AIO15)
AIOMUX2	0x6FB8	2	Analog IO MUX 2 Register (AIO16 to AIO31)
AIODIR	0x6FBA	2	Analog IO Direction Register (AIO0 AIO31)

(1) An X in a table cell indicates the bit can be 0 or 1.

Table 4-38. GPIO Trip Input Select Registers

Name[1]	Address	Size (x16)	Description
GPTRIP1SEL	0x5FE0	1	GPTRIP1 (TZ1n) Input Select Register (GPIO0 - GPIO63)
GPTRIP2SEL	0x5FE1	1	GPTRIP2 (TZ2n, ADCEXTTRIG) Input Select Register (GPIO0 - GPIO63)
GPTRIP3SEL	0x5FE2	1	GPTRIP3 (TZ3n) Input Select Register (GPIO0 - GPIO63)
GPTRIP4SEL	0x5FE3	1	GPTRIP4 (XINT1) Input Select Register (GPIO0 - GPIO63)

(1) 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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-38. GPIO Trip Input Select Registers (continued)

Name[1]	Address	Size (x16)	Description
GPTRIP5SEL	0x5FE4	1	GPTRIP5 (XINT2) Input Select Register (GPIO0 - GPIO63)
GPTRIP6SEL	0x5FE5	1	GPTRIP6 (XINT3) Input Select Register (GPIO0 - GPIO63)
GPTRIP7SEL	0x5FE6	1	GPTRIP7 (ECAP1) Input Select Register (GPIO0 - GPIO63)
GPTRIP8SEL	0x5FE7	1	GPTRIP8 (ECAP2) Input Select Register (GPIO0 - GPIO63)
GPIOLPMSEL1	0x5FE8	2	LPM GPIO Select 1 Register (GPIO0 - GPIO31)
GPIOLPMSEL2	0x5FEA	2	LPM GPIO Select 2 Register (GPIO32 - GPIO63)
GPTRIP9SEL	0x5FF0	1	GPTRIP9 (ECAP3) Input Select Register (GPIO0 - GPIO63)
GPTRIP10SEL	0x5FF1	1	GPTRIP10 (ECAP4) Input Select Register (GPIO0 - GPIO63)
GPTRIP11SEL	0x5FF2	1	GPTRIP11 (ECAP5) Input Select Register (GPIO0 - GPIO63)
GPTRIP12SEL	0x5FF3	1	GPTRIP12 (ECAP6) Input Select Register (GPIO0 - GPIO63)

To plan configuration of the GPIO module, consider the following steps:

1. **Plan the device pin-out:**

   Through a pin multiplexing scheme, a lot of flexibility is provided for assigning functionality to the GPIO-capable pins. Before getting started, look at the peripheral options available for each pin, and plan pin-out for your specific system. Will the pin be used as a general purpose input or output (GPIO) or as one of up to three available peripheral functions? Knowing this information will help determine how to further configure the pin.

2. **Select if the GPIO will be C28 core controlled:**

   If the pin will be used as a C28 GPIO or peripheral, the correct bits must be set in the GPIOCSEL register. This register is located in the M3 GPIO register space.

3. **Enable or disable internal pull-up resistors:**

   To enable or disable the internal pullup resistors, write to the respective bits in the GPIO pullup disable (GPIOPUR) register. This register is located in the M3 GPIO register space. All GPIO-capable pins have the pullup disabled by default. The AIOx pins do not have internal pull-up resistors.

4. **Select input qualification:**

   If the pin will be used as an input, specify the required input qualification, if any. The input qualification is specified in the GPACTRL, GPBCTRL, GPCCTRL, GPECTRL, GPAQSEL1, GPAQSEL2, GPBQSEL1, GPBQSEL2, GPCQSEL1, and GPEQSEL1 registers. By default, GPIO Port A, B, and C are synchronized to SYSCLKOUT only, and GPIO Port E is synchronized to the analog subsystem clock.

5. **Select the pin function:**

   Configure the GPxMUXn or AIOMUXn registers such that the pin is a GPIO or one of three available peripheral functions. By default, all GPIO-capable pins are configured at reset as general purpose input pins.

6. **For digital general purpose I/O, select the direction of the pin:**

   If the pin is configured as an GPIO, specify the direction of the pin as either input or output in the GPADIR, GPBDIR, GPCDIR, GPEDIR, or AIODIR registers. By default, all GPIO pins are inputs. To change the pin from input to output, first load the output latch with the value to be driven by writing the appropriate value to the GPxCLEAR, GPxSET, or GPxTOGGLE (or AIOCLEAR, AIOSET, or AIOTOGGLE) registers. Once the output latch is loaded, change the pin direction from input to output via the GPxDIR registers. The output latch for all pins is cleared at reset.

7. **Select low power mode wake-up sources:**

Copyright © 2012–2019, Texas Instruments Incorporated

Specify which pins, if any, will be able to wake the device from HALT and STANDBY low power modes. The pins are specified in the GPIOLPMSEL1 and GPIOLPMSEL2 registers.

8. **Select external interrupt sources:**

   Specify the source for the XINT1 - XINT3 interrupts. For each interrupt you can specify one of the GPIO (0-63) signals as the source. This is done by setting the correct bits in GPTRIP4SEL for XINT1, GPTRIP5SEL for XINT2, and GPTRIP6SEL for XINT3. The polarity of the interrupts can be configured in the XINTnCR register.

NOTE: There is a 2-SYSCLKOUT cycle delay from when a write to configuration registers such as GPxMUXn and GPxQSELn occurs to when the action is valid

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.4 Digital General Purpose I/O Control

For pins that are configured as GPIO you can change the values on the pins by using the registers in Table 4-39.

Table 4-39. GPIO Data Registers

Name	Address	Size (x16)	Register Description
GPADAT	0x5FC0	4	GPIO A Data Register (GPIO0 - GPIO31)
GPASET	0x5FC2	4	GPIO A Set Register (GPIO0 - GPIO31)
GPACLEAR	0x5FC4	4	GPIO A Clear Register (GPIO0 - GPIO31)
GPATOGGLE	0x5FC6	4	GPIO A Toggle Register (GPIO0 - GPIO31)
GPBDAT	0x5FC8	4	GPIO B Data Register (GPIO32 - GPIO63)
GPBSET	0x5FCA	4	GPIO B Set Register (GPIO32 - GPIO63)
GPBCLEAR	0x5FCC	4	GPIO B Clear Register (GPIO32 - GPIO63)
GPBTOGGLE	0x5FCE	4	GPIO B Toggle Register (GPIO32 - GPIO63)
GPCDAT	0x5FD0	4	GPIO C Data Register (GPIO68 - GPIO71)
GPCSET	0x5FD2	4	GPIO C Set Register (GPIO68 - GPIO71)
GPCCLEAR	0x5FD4	4	GPIO C Clear Register (GPIO68 - GPIO71)
GPCTOGGLE	0x5FD6	4	GPIO C Toggle Register (GPIO68 - GPIO71)
GPEDAT	0x6FC0	4	GPIO E Data Register (GPIO128 - GPIO135)
GPESET	0x6FC2	4	GPIO E Set Register (GPIO128 - GPIO135)
GPECLEAR	0x6FC4	4	GPIO E Clear Register (GPIO128 - GPIO135)
GPETOGGLE	0x6FC6	4	GPIO E Toggle Register (GPIO128 - GPIO135)
AIODAT	0x6FD8	4	Analog IO Data Register (AIO0 - AIO31)
AIOSET	0x6FDA	4	Analog IO Set Register (AIO0 - AIO31)
AIOCLEAR	0x6FDC	4	Analog IO Clear Register (AIO0 - AIO31)
AIOTOGGLE	0x6FDE	4	Analog IO Toggle Register (AIO0 - AIO31)

- **GPxDAT/AIODAT Registers**

  Each I/O port has one data register. Each bit in the data register corresponds to one GPIO pin. No matter how the pin is configured (GPIO or peripheral function), the corresponding bit in the data register reflects the current state of the pin after qualification (This does not apply to AIOx pins). Writing to the GPxDAT/AIODAT register clears or sets the corresponding output latch and if the pin is enabled as a general purpose output (GPIO output) the pin will also be driven either low or high. If the pin is not configured as a GPIO output then the value will be latched, but the pin will not be driven. Only if the pin is later configured as a GPIO output, will the latched value be driven onto the pin.

  When using the GPxDAT register to change the level of an output pin, you should be cautious not to

Copyright © 2012–2019, Texas Instruments Incorporated

accidentally change the level of another pin. For example, if you mean to change the output latch level of GPIOA1 by writing to the GPADAT register bit 0 using a read-modify-write instruction, a problem can occur if another I/O port A signal changes level between the read and the write stage of the instruction. Following is an analysis of why this happens:

The GPxDAT registers reflect the state of the pin, not the latch. This means the register reflects the actual pin value. However, there is a lag between when the register is written to when the new pin value is reflected back in the register. This may pose a problem when this register is used in subsequent program statements to alter the state of GPIO pins. An example is shown below where two program statements attempt to drive two different GPIO pins that are currently low to a high state.

If Read-Modify-Write operations are used on the GPxDAT registers, because of the delay between the output and the input of the first instruction (I1), the second instruction (I2) will read the old value and write it back.

```
GpioDataRegs.GPADAT.bit.GPIO1 = 1 ;
I1 performs read-modify-write of GPADAT GpioDataRegs.GPADAT.bit.GPIO2 = 1 ;
I2 also a read-modify-write of GPADAT. ;
It gets the old value of GPIO1 due to the delay
```

The second instruction will wait for the first to finish its write due to the write-followed-by-read protection on this peripheral frame. There will be some lag, however, between the write of (I1) and the GPxDAT bit reflecting the new value (1) on the pin. During this lag, the second instruction will read the old value of GPIO1 (0) and write it back along with the new value of GPIO2 (1). Therefore, GPIO1 pin stays low.

One solution is to put some NOP's between instructions. A better solution is to use the GPxSET/GPxCLEAR/GPxTOGGLE registers instead of the GPxDAT registers. These registers always read back a 0 and writes of 0 have no effect. Only bits that need to be changed can be specified without disturbing any other bit(s) that are currently in the process of changing.

- **GPxSET/AIOSET Registers**

  The set registers are used to drive specified GPIO pins high without disturbing other pins. Each I/O port has one set register and each bit corresponds to one GPIO pin. The set registers always read back 0. If the corresponding pin is configured as an output, then writing a 1 to that bit in the set register will set the output latch high and the corresponding pin will be driven high. If the pin is not configured as a GPIO output, then the value will be latched but the pin will not be driven. Only if the pin is later configured as a GPIO output will the latched value will be driven onto the pin. Writing a 0 to any bit in the set registers has no effect.

- **GPxCLEAR/AIOCLEAR Registers**

  The clear registers are used to drive specified GPIO pins low without disturbing other pins. Each I/O port has one clear register. The clear registers always read back 0. If the corresponding pin is configured as a general purpose output, then writing a 1 to the corresponding bit in the clear register will clear the output latch and the pin will be driven low. If the pin is not configured as a GPIO output, then the value will be latched but the pin will not be driven. Only if the pin is later configured as a GPIO output will the latched value will be driven onto the pin. Writing a 0 to any bit in the clear registers has no effect.

- **GPxTOGGLE/AIOTOGGLE Registers**

  The toggle registers are used to drive specified GPIO pins to the opposite level without disturbing other pins. Each I/O port has one toggle register. The toggle registers always read back 0. If the corresponding pin is configured as an output, then writing a 1 to that bit in the toggle register flips the output latch and pulls the corresponding pin in the opposite direction. That is, if the output pin is driven low, then writing a 1 to the corresponding bit in the toggle register will pull the pin high. Likewise, if the output pin is high, then writing a 1 to the corresponding bit in the toggle register will pull the pin low. If the pin is not configured as a GPIO output, then the value will be latched but the pin will not be driven. Only if the pin is later configured as a GPIO output will the latched value will be driven onto the pin. Writing a 0 to any bit in the toggle registers has no effect.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.5 Input Qualification

The input qualification scheme has been designed to be very flexible. You can select the type of input qualification for each GPIO pin by configuring the GPxQSEL1 and GPxQSEL2 registers. In the case of a GPIO input pin, the qualification can be specified as only synchronize to SYSCLKOUT or qualification by a sampling window. For pins that are configured as peripheral inputs, the input can also be asynchronous in addition to synchronized to SYSCLKOUT or qualified by a sampling window. The remainder of this section describes the options available.

4.2.5.1 No Synchronization (asynchronous input)

This mode is used for peripherals where input synchronization is not required or the peripheral itself performs the synchronization. Examples include communication ports SCI, SPI and I^2C. In addition, it may be desirable to have the ePWM trip zone ($\overline{TZn}$) signals function independent of the presence of SYSCLKOUT.

The asynchronous option is not valid if the pin is used as a general purpose digital input pin (GPIO). If the pin is configured as a GPIO input and the asynchronous option is selected then the qualification defaults to synchronization to SYSCLKOUT as described in Section 4.2.5.2.

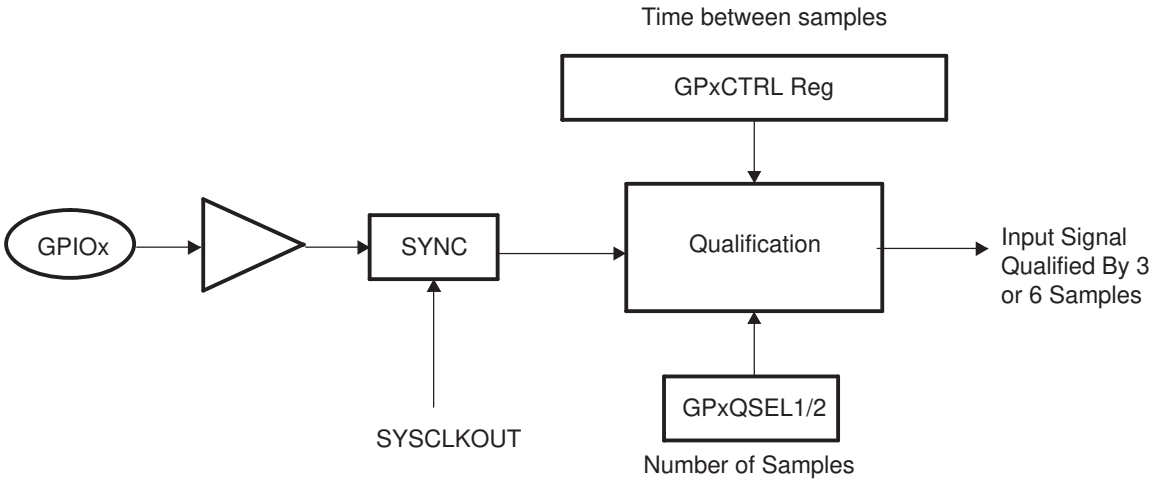
4.2.5.2 Synchronization to SYSCLKOUT Only

This is the default qualification mode of all the pins on GPIO Port A, B, and C. The GPIO pins on Port E are synchronized to the analog subsystem clock by default. In this mode, the input signal is only synchronized to the system clock (SYSCLKOUT). Because the incoming signal is asynchronous, it can take up to a SYSCLKOUT period of delay in order for the input to the DSP to be changed. No further qualification is performed on the signal.

4.2.5.3 Qualification Using a Sampling Window

In this mode, the signal is first synchronized to the system clock (SYSCLKOUT) and then qualified by a specified number of cycles before the input is allowed to change. Figure 4-40 and Figure 4-41 show how the input qualification is performed to eliminate unwanted noise. Two parameters are specified by the user for this type of qualification: 1) the sampling period, or how often the signal is sampled, and 2) the number of samples to be taken.

Figure 4-40. Input Qualification Using a Sampling Window



Time between samples (sampling period):

To qualify the signal, the input signal is sampled at a regular period. The sampling period is specified by the user and determines the time duration between samples, or how often the signal will be sampled, relative to the CPU clock (SYSCLKOUT).

Copyright © 2012–2019, Texas Instruments Incorporated

The sampling period is specified by the qualification period (QUALPRDn) bits in the GPxCTRL register. The sampling period is configurable in groups of 8 input signals. For example, GPIO0 to GPIO7 use GPACTRL[QUALPRD0] setting and GPIO8 to GPIO15 use GPACTRL[QUALPRD1]. Table 4-40 and Table 4-41 show the relationship between the sampling period or sampling frequency and the GPxCTRL[QUALPRDn] setting.

Table 4-40. Sampling Period

	Sampling Period
If GPxCTRL[QUALPRDn] = 0	$1 \times T_{SYSCLKOUT}$
If GPxCTRL[QUALPRDn] ≠ 0	$2 \times$ GPxCTRL[QUALPRDn] $\times T_{SYSCLKOUT}$
	Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT

Table 4-41. Sampling Frequency

	Sampling Frequency
If GPxCTRL[QUALPRDn] = 0	$f_{SYSCLKOUT}$
If GPxCTRL[QUALPRDn] ≠ 0	$f_{SYSCLKOUT} \times 1 \div (2 \times$ GPxCTRL[QUALPRDn])
	Where $f_{SYSCLKOUT}$ is the frequency of SYSCLKOUT

From these equations, the minimum and maximum time between samples can be calculated for a given SYSCLKOUT frequency:

Example: Maximum Sampling Frequency:

If GPxCTRL[QUALPRDn] = 0
then the sampling frequency is $f_{SYSCLKOUT}$
If, for example, $f_{SYSCLKOUT}$ = 150 MHz
then the signal will be sampled at 150 MHz or one sample every 6.67 ns.

Example: Minimum Sampling Frequency:

If GPxCTRL[QUALPRDn] = 0xFF (i.e. 255)
then the sampling frequency is $f_{SYSCLKOUT} \times 1 \div (2 \times$ GPxCTRL[QUALPRDn])
If, for example, $f_{SYSCLKOUT}$ = 150 MHz
then the signal will be sampled at 150 MHz $\times 1 \div (2 \times 255)$ or one sample every 3.4 μs.

Number of samples:

The number of times the signal is sampled is either three samples or six samples as specified in the qualification selection (GPxQSEL1 and GPxQSEL2, registers). When three or six consecutive cycles are the same, then the input change will be passed through to the DSP.

Total Sampling Window Width:

The sampling window is the time during which the input signal will be sampled as shown in Figure 4-41. By using the equation for the sampling period along with the number of samples to be taken, the total width of the window can be determined.

For the input qualifier to detect a change in the input, the level of the signal must be stable for the duration of the sampling window width or longer.

The number of sampling periods within the window is always one less then the number of samples taken. For a thee-sample window, the sampling window width is 2 sampling periods wide where the sampling period is defined in Table 4-40. Likewise, for a six-sample window, the sampling window width is 5 sampling periods wide. Table 4-42 and Table 4-43 show the calculations that can be used to determine the total sampling window width based on GPxCTRL[QUALPRDn] and the number of samples taken.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-42. Case 1: Three-Sample Sampling Window Width

	Total Sampling Window Width
If GPxCTRL[QUALPRDn] = 0	2 × $T_{SYSCLKOUT}$
If GPxCTRL[QUALPRDn] ≠ 0	2 × 2 × GPxCTRL[QUALPRDn] × $T_{SYSCLKOUT}$
	Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT

Table 4-43. Case 2: Six-Sample Sampling Window Width

	Total Sampling Window Width
If GPxCTRL[QUALPRDn] = 0	5 × $T_{SYSCLKOUT}$
If GPxCTRL[QUALPRDn] ≠ 0	5 × 2 × GPxCTRL[QUALPRDn] × $T_{SYSCLKOUT}$
	Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT

NOTE: The external signal change is asynchronous with respect to both the sampling period and SYSCLKOUT. Due to the asynchronous nature of the external signal, the input should be held stable for a time greater than the sampling window width to make sure the logic detects a change in the signal. The extra time required can be up to an additional sampling period + $T_{SYSCLKOUT}$.

The required duration for an input signal to be stable for the qualification logic to detect a change is described in the device specific data manual.

Example Qualification Window:

For the example shown in Figure 4-41, the input qualification has been configured as follows:

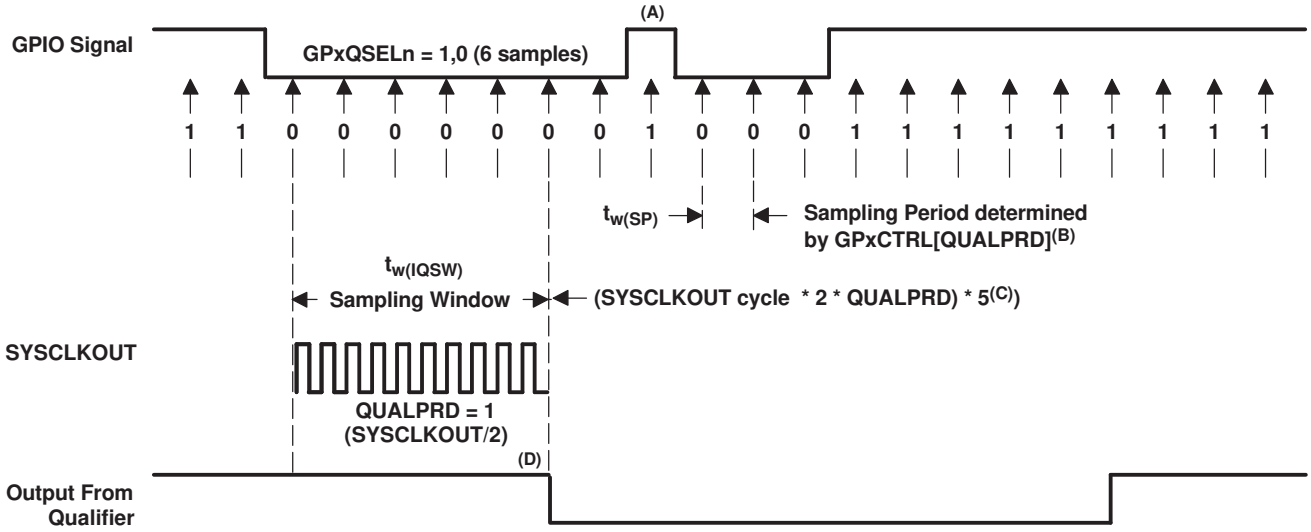
- GPxQSEL1/2 = 1,0. This indicates a six-sample qualification.
- GPxCTRL[QUALPRDn] = 1. The sampling period is $t_w(SP)$ = 2 × GPxCTRL[QUALPRDn] × $T_{SYSCLKOUT}$.

This configuration results in the following:

- The width of the sampling window is: .
  $t_w(IQSW) = 5 \times t_w(SP) = 5 \times 2 \times$ GPxCTRL[QUALPRDn] × $T_{SYSCLKOUT}$ or 5 × 2 × $T_{SYSCLKOUT}$
- If, for example, $T_{SYSCLKOUT}$ = 6.67 ns, then the duration of the sampling window is:
  $t_w(IQSW)$ = 5 × 2 × 6.67 ns =66.7 ns.
- To account for the asynchronous nature of the input relative to the sampling period and SYSCLKOUT, up to an additional sampling period, $t_w(SP)$, + $T_{SYSCLKOUT}$ may be required to detect a change in the input signal. For this example:
  $t_w(SP) + T_{SYSCLKOUT}$ = 333.4 ns + 66.67 ns = 400.1 ns
- In Figure 4-41, the glitch (A) is shorter then the qualification window and will be ignored by the input qualifier.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-41. Input Qualifier Clock Cycles



A. This glitch will be ignored by the input qualifier. The QUALPRD bit field specifies the qualification sampling period. It can vary from 00 to 0xFF. If QUALPRD = 00, then the sampling period is 1 SYSCLKOUT cycle. For any other value “n”, the qualification sampling period in 2n SYSCLKOUT cycles (i.e., at every 2n SYSCLKOUT cycles, the GPIO pin will be sampled).
B. The qualification period selected via the GPxCTRL register applies to groups of 8 GPIO pins.
C. The qualification block can take either three or six samples. The GPxQSELn Register selects which sample mode is used.
D. In the example shown, for the qualifier to detect the change, the input should be stable for 10 SYSCLKOUT cycles or greater. In other words, the inputs should be stable for (5 x QUALPRD x 2) SYSCLKOUT cycles. That would ensure 5 sampling periods for detection to occur. Since external signals are driven asynchronously, an 13-SYSCLKOUT-wide pulse ensures reliable recognition.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.6 GPIO and Peripheral Multiplexing (MUX)

Up to three different peripheral functions are multiplexed along with a general input/output (GPIO) function per pin. This allows you to pick and choose a peripheral mix that will work best for the particular application.

Table 4-45 and Table 4-46 show an overview of the possible multiplexing combinations sorted by GPIO pin. The second column indicates the I/O name of the pin on the device. Since the I/O name is unique, it is the best way to identify a particular pin. Therefore, the register descriptions in this section only refer to the GPIO name of a particular pin. The MUX register and particular bits that control the selection for each pin are indicated in the first column.

For example, the multiplexing for the GPIO6 pin is controlled by writing to GPAMUX[13:12]. By writing to these bits, the pin is configured as either GPIO6, or one of up to three peripheral functions. The GPIO6 pin can be configured as follows:

GPAMUX1[13:12] Bit Setting	Pin Functionality Selected
If GPAMUX1[13:12] = 0,0	Pin configured as GPIO6
If GPAMUX1[13:12] = 0,1	Pin configured as EPWM4A (O)
If GPAMUX1[13:12] = 1,0	Reserved
If GPAMUX1[13:12] = 1,1	Pin configured as EPWMSYNCO (O)

NOTE: If you should select a reserved GPIO MUX configuration that is not mapped to a peripheral, the state of the pin will be undefined and the pin may be driven. Reserved configurations are for future expansion and should not be selected. In the device MUX tables, these options are indicated as Reserved .

Some peripherals can be assigned to more than one pin via the MUX registers. For example, the ECAP1 can be assigned to either the GPIO5 or GPIO24 pin, depending on individual system requirements as shown below:

Signal Pin Assigned to SPISIMOB		MUX Configuration
Choice 1	GPIO5	GPAMUX[11:10] = 1,1
or Choice 2	GPIO24	GPAMUX2[17:16] = 0,1

If no pin is configured as an input to a peripheral, or if more than one pin is configured as an input for the same peripheral, then the input to the peripheral will either default to a 0 or a 1 as shown in Table 4-44. For example, if ECAP1 was assigned to both GPIO5 and GPIO24, the input to the ECAP1 peripheral would default to a high state as shown in Table 4-44 and the input would not be connected to GPIO5 or GPIO24.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-44. Default State of Peripheral Input

Peripheral Input	Description	Default Input [1]
TZ1-TZ3	Trip zone 1-3	1
EPWMSYNCI	ePWM Synch Input	0
ECAP1	eCAP1 input	1
EQEP1A	eQEP input	1
EQEP1I	eQEP index	1
EQEP1S	eQEP strobe	1
SPICLKA/SPICLKB	SPI-A clock	1
SPISTEA /SPISTEB	SPI-A transmit enable	0
SPISIMOA/SPISIMOB	SPI-A Slave-in, master-out	1
SPISOMIA/SPISOMIB	SPI-A Slave-out, master-in	1
SCIRXDA - SCIRXDB	SCI-A - SCI-B receive	1
SDAA	I^2C data	1
SCLA1	I^2C clock	1

(1) This value will be assigned to the peripheral input if more then one pin has been assigned to the peripheral function in the GPxMUX1/2 registers or if no pin has been assigned.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-45. GPIOA MUX

	Default at Reset Primary I/O Function	Peripheral Selection	Peripheral Selection 2	Peripheral Selection 3
GPAMUX1 Register Bits	**(GPAMUX1 bits = 00)**	**(GPAMUX1 bits = 01)**	**(GPAMUX1 bits = 10)**	**(GPAMUX1 bits = 11)**
1-0	GPIO0	EPWM1A (O)	Reserved	Reserved
3-2	GPIO1	EPWM1B (O)	ECAP6 (I/O)	Reserved
5-4	GPIO2	EPWM2A (O)	Reserved	Reserved
7-6	GPIO3	EPWM2B (O)	ECAP5 (I/O)	Reserved
9-8	GPIO4	EPWM3A (O)	Reserved	Reserved
11-10	GPIO5	EPWM3B (O)	MFSRA (I/O)	ECAP1 (O)
13-12	GPIO6	EPWM4A (O)	Reserved	EPWMSYNCO (O)
15-14	GPIO7	EPWM4B (O)	MCLKRA (I/O)	ECAP2 (O)
17-16	GPIO8	EPWM5A (O)	Reserved	$\overline{\text{ADCSOCAO}}$ (O)
19-18	GPIO9	EPWM5B (O)	Reserved	ECAP3 (O)
21-20	GPIO10	EPWM6A (O)	Reserved	$\overline{\text{ADCSOCBO}}$ (O)
23-22	GPIO11	EPWM6B (O)	Reserved	ECAP4 (O)
25-24	GPIO12	EPWM7A (O)	Reserved	Reserved
27-26	GPIO13	EPWM7B (O)	Reserved	Reserved
29-28	GPIO14	EPWM8A (O)	Reserved	Reserved
31-30	GPIO15	EPWM8B (O)	Reserved	Reserved
GPAMUX2 Register Bits	**(GPAMUX2 bits = 00)**	**(GPAMUX2 bits = 01)**	**(GPAMUX2 bits = 10)**	**(GPAMUX2 bits = 11)**
1-0	GPIO16	SPISIMOA (I/O)	Reserved	Reserved
3-2	GPIO17	SPISOMIA (I/O)	Reserved	Reserved
5-4	GPIO18	SPICLKA (I/O)	Reserved	Reserved
7-6	GPIO19	$\overline{\text{SPISTEA}}$ (I/O)	Reserved	Reserved
9-8	GPIO20	EQEP1A (I)	MDXA (O)	Reserved
11-10	GPIO21	EQEP1B (I)	MDRA (I)	Reserved
13-12	GPIO22	EQEP1S (I/O)	MCLKXA (I/O)	Reserved
15-14	GPIO23	EQEP1I (I/O)	MFSXA (I/O)	Reserved
17-16	GPIO24	ECAP1 (I/O)	EQEP2A (I)	Reserved
19-18	GPIO25	ECAP2 (I/O)	EQEP2B (I)	Reserved
21-20	GPIO26	ECAP3 (I/O)	EQEP2I (I/O)	Reserved
23-22	GPIO27	ECAP4 (I/O)	EQEP2S (I/O)	Reserved
25-24	GPIO28	SCIRXDA (I)	Reserved	Reserved
27-26	GPIO29	SCITXDA (O)	Reserved	Reserved
29-28	GPIO30	Reserved	Reserved	EPWM9A (O)
31-30	GPIO31	Reserved	Reserved	EPWM9B (O)

Note: The word Reserved means that there is no peripheral assigned to this GPxMUX1/2 register setting. Should it be selected, the state of the pin will be undefined and the pin may be driven. This selection is a reserved configuration for future expansion.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-46. GPIOB MUX

	Default at Reset Primary I/O Function	Peripheral Selection 1	Peripheral Selection 2	Peripheral Selection 3
GPBMUX1 Register Bits	**(GPBMUX1 bits = 00)**	**(GPBMUX1 bits = 01)**	**(GPBMUX1 bits = 10)**	**(GPBMUX1 bits = 11)**
1-0	GPIO32	SDAA (I/OC)	SCIRXDA (I)	$\overline{\text{ADCSOCAO}}$ (O)
3-2	GPIO33	SCLA (I/OC)	EPWMSYNCO (O)	$\overline{\text{ADCSOCBO}}$ (O)
5-4	GPIO34	ECAP1 (I/O)	SCIRXDA (I)	XCLKOUT (O)
7-6	GPIO35	SCITXDA (O)	Reserved	Reserved
9-8	GPIO36	SCIRXDA (I)	Reserved	Reserved
11-10	GPIO37	ECAP2 (I/O)	Reserved	Reserved
13-12	GPIO38	Reserved	Reserved	Reserved
15-14	GPIO39	Reserved	Reserved	Reserved
17-16	GPIO40	Reserved	Reserved	Reserved
19-18	GPIO41	Reserved	Reserved	Reserved
21-20	GPIO42	Reserved	Reserved	Reserved
23-22	GPIO43	Reserved	Reserved	Reserved
25-24	GPIO44	Reserved	Reserved	Reserved
27-26	GPIO45	Reserved	Reserved	Reserved
29-28	GPIO46	Reserved	Reserved	Reserved
31-30	GPIO47	Reserved	Reserved	Reserved
GPBMUX2 Register Bits	**(GPBMUX2 bits = 00)**	**(GPBMUX2 bits = 01)**	**(GPBMUX2 bits = 10)**	**(GPBMUX2 bits = 11)**
1-0	GPIO48	ECAP5 (I/O)	Reserved	Reserved
3-2	GPIO49	ECAP6 (I/O)	Reserved	Reserved
5-4	GPIO50	EQEP1A (I)	Reserved	Reserved
7-6	GPIO51	EQEP1B (I)	Reserved	Reserved
9-8	GPIO52	EQEP1S (I/O)	Reserved	Reserved
11-10	GPIO53	EQEP1I (I/O)	Reserved	Reserved
13-12	GPIO54	SPISIMOA (I/O)	Reserved	EQEP3A (I)
15-14	GPIO55	SPISOMIA (I/O)	Reserved	EQEP3B (I)
17-16	GPIO56	SPICLKA (I/O)	Reserved	EQEP3S (I/O)
19-18	GPIO57	SPISTEA (I/O)	Reserved	EQEP3I (I/O)
21-20	GPIO58	MCLKRA (I/O)	Reserved	EPWM7A (O)
23-22	GPIO59	MFSRA (I/O)	Reserved	EPWM7B (O)
25-24	GPIO60	Reserved	Reserved	EPWM8A (O)
27-26	GPIO61	Reserved	Reserved	EPWM8B (O)
29-28	GPIO62	Reserved	Reserved	EPWM9A(O)
31-30	GPIO63/XCLKIN	Reserved	Reserved	EPWM9B (O)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-47. GPIOC MUX

	Default at Reset Primary I/O Function	Peripheral Selection 1	Peripheral Selection 2	Peripheral Selection 3
GPCMUX1 Register Bits	**(GPCMUX1 bits = 00)**	**(GPCMUX1 bits = 01)**	**(GPCMUX1 bits = 10)**	**(GPCMUX1 bits = 11)**
1-0	Not supported	Reserved	Reserved	Reserved
3-2	Not supported	Reserved	Reserved	Reserved
5-4	Not supported	Reserved	Reserved	Reserved
7-6	Not supported	Reserved	Reserved	Reserved
9-8	GPIO68	Reserved	Reserved	Reserved
11-10	GPIO69	Reserved	Reserved	Reserved
13-12	GPIO70	Reserved	Reserved	Reserved
15-14	Not supported	Reserved	Reserved	Reserved
17-16	Not supported	Reserved	Reserved	Reserved
19-18	Not supported	Reserved	Reserved	Reserved
21-20	Not supported	Reserved	Reserved	Reserved
23-22	Not supported	Reserved	Reserved	Reserved
25-24	Not supported	Reserved	Reserved	Reserved
27-26	Not supported	Reserved	Reserved	Reserved
29-28	Not supported	Reserved	Reserved	Reserved
31-30	Not supported	Reserved	Reserved	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-48. GPIOE MUX

	Default at Reset Primary I/O Function	Peripheral Selection 1	Peripheral Selection 2	Peripheral Selection 3
GPEMUX1 Register Bits	(GPEMUX1 bits = 00)	(GPEMUX1 bits = 01)	(GPEMUX1 bits = 10)	(GPEMUX1 bits = 11)
1-0	GPIO128	Reserved	Reserved	Reserved
3-2	GPIO129	Reserved	Reserved	COMP1OUT (O)
5-4	GPIO130	Reserved	Reserved	COMP6OUT (O)
7-6	GPIO131	Reserved	Reserved	COMP2OUT (O)
9-8	GPIO132	Reserved	Reserved	COMP3OUT (O)
11-10	GPIO133	Reserved	Reserved	COMP4OUT (O)
13-12	GPIO134	Reserved	Reserved	Reserved
15-14	GPIO135	Reserved	Reserved	COMP5OUT (O)

Table 4-49. Analog MUX

	AIOx and Peripheral Selection1	Default at Reset Peripheral Selection 2 and Peripheral Selection 3
AIOMUX1 Register bits	AIOMUX1 bits = 0,x	AIOMUX1 bits = 1,x
1-0	ADCINA0 (I)	ADCINA0 (I)
3-2	ADCINA1 (I)	ADCINA1 (I)
5-4	AIO2 (I/O)	ADCINA2 (I), COMP1A (I)
7-6	ADCINA3 (I)	ADCINA3 (I)
9-8	AIO4 (I/O)	ADCINA4 (I), COMP2A (I)
11-10	ADCINA5 (I)	ADCINA5 (I)
13-12	AIO6 (I/O)	ADCINA6 (I), COMP3A (1)
15-14	ADCINA7 (I)	ADCINA7 (I)
17-16	ADCINB0 (I)	ADCINB0 (I)
19-18	ADCINB1 (I)	ADCINB1 (I)
21-20	AIO10 (I/O)	ADCINB2 (I), COMP1B (I)
23-22	ADCINB3 (I)	ADCINB3 (I)
25-24	AIO12 (I/O)	ADCINB4 (I), COMP2B (I)
27-26	ADCINB5 (I)	ADCINB5 (I)
29-28	AIO14 (I/O)	ADCINB6 (I), COMP3B (1)
31-30	ADCINB7 (I)	ADCINB7 (I)
AIOMUX2 Register bits	**AIOMUX2 bits = 0,x**	**AIOMUX2 bits = 1,x**
1-0	ADCINA0 (I)	ADCINA0 (I)
3-2	ADCINA1 (I)	ADCINA1 (I)
5-4	AIO18 (I/O)	ADCINA2 (I), COMP4A (I)
7-6	ADCINA3 (I)	ADCINA3 (I)
9-8	AIO20 (I/O)	ADCINA4 (I), COMP5A (I)
11-10	ADCINA5 (I)	ADCINA5 (I)
13-12	AIO22 (I/O)	ADCINA6 (I), COMP6A (1)
15-14	ADCINA7 (I)	ADCINA7 (I)
17-16	ADCINB0 (I)	ADCINB0 (I)
19-18	ADCINB1 (I)	ADCINB1 (I)
21-20	AIO26 (I/O)	ADCINB2 (I), COMP4B (I)
23-22	ADCINB3 (I)	ADCINB3 (I)
25-24	AIO28 (I/O)	ADCINB4 (I), COMP5B (I)
27-26	ADCINB5 (I)	ADCINB5 (I)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-49. Analog MUX (continued)

		Default at Reset
	AIOx and Peripheral Selection1	Peripheral Selection 2 and Peripheral Selection 3
AIOMUX1 Register bits	AIOMUX1 bits = 0,x	AIOMUX1 bits = 1,x
29-28	AIO30 (I/O)	ADCINB6 (I), COMP6B (1)
31-30	ADCINB7 (I)	ADCINB7 (I)

Note: AIOMUX1 registers correspond to ADC1 and AIOMUX2 registers correspond to ADC2.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7 Register Bit Definitions

The following sections provide descriptive information on the C28 registers.

4.2.7.1 GPIO Port A MUX 1 (GPAMUX1) Register

The GPIO Port A MUX 1 (GPAMUX1) register is shown and described in the figure and table below.

Figure 4-42. GPIO Port A MUX 1 (GPAMUX1) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND- R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-50. GPIO Port A Multiplexing 1 (GPAMUX1) Register Field Descriptions

Bits	Field	Value	Description [(1)]
31-30	GPIO15		Configure the GPIO15 pin as:
		00	GPIO15 - General purpose input/output 15 (default) (I/O)
		01	EPWM8B - ePWM8 output B (O)
		10	Reserved
		11	Reserved
29-28	GPIO14		Configure the GPIO14 pin as:
		00	GPIO14 - General purpose I/O 14 (default) (I/O)
		01	EPWM8A - ePWM8 output A (O)
		10	Reserved
		11	Reserved
27-26	GPIO13		Configure the GPIO13 pin as:
		00	GPIO13 - General purpose I/O 13 (default) (I/O)
		01	EPWM7B - ePWM7 output B (O)
		10	Reserved
		11	Reserved
25-24	GPIO12		Configure the GPIO12 pin as:
		00	GPIO12 - General purpose I/O 12 (default) (I/O)
		01	EPWM7A - ePWM7 output A (O)
		10	Reserved
		11	Reserved
23-22	GPIO11		Configure the GPIO11 pin as:
		00	GPIO11 - General purpose I/O 11 (default) (I/O)
		01	EPWM6B - ePWM6 output B (O)
		10	Reserved
		11	ECAP4 - eCAP4 (I/O)
21-20	GPIO10		Configure the GPIO10 pin as:
		00	GPIO10 - General purpose I/O 10 (default) (I/O)
		01	EPWM6A - ePWM6 output A (O)
		10	Reserved
		11	$\overline{\text{ADCSOCBO}}$ - ADC Start of conversion B (O)

(1) This register is EALLOW protected.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-50. GPIO Port A Multiplexing 1 (GPAMUX1) Register Field Descriptions (continued)

Bits	Field	Value	Description [1]
19-18	GPIO9		Configure the GPIO9 pin as:
		00	GPIO9 - General purpose I/O 9 (default) (I/O)
		01	EPWM5B - ePWM5 output B
		10	Reserved
		11	ECAP3 - eCAP3 (I/O)
17-16	GPIO8		Configure the GPIO8 pin as:
		00	GPIO8 - General purpose I/O 8 (default) (I/O)
		01	EPWM5A - ePWM5 output A (O)
		10	Reserved
		11	$\overline{\text{ADCSOCAO}}$ - ADC Start of conversion A
15-14	GPIO7		Configure the GPIO7 pin as:
		00	GPIO7 - General purpose I/O 7 (default) (I/O)
		01	EPWM4B - ePWM4 output B (O)
		10	MCLKRA - McBSP - A receive clock (I/O)
		11	ECAP2 - eCAP2 (I/O)
13-12	GPIO6		Configure the GPIO6 pin as:
		00	GPIO6 - General purpose I/O 6 (default)
		01	EPWM4A - ePWM4 output A (O)
		10	Reserved
		11	EPWMSYNCO - ePWM Synch-out (O)
11-10	GPIO5		Configure the GPIO5 pin as:
		00	GPIO5 - General purpose I/O 5 (default) (I/O)
		01	EPWM3B - ePWM3 output B
		10	MFSRA - McBSP - A receive frame synch (I/O)
		11	ECAP1 - eCAP1 (I/O)
9-8	GPIO4		Configure the GPIO4 pin as:
		00	GPIO4 - General purpose I/O 4 (default) (I/O)
		01	EPWM3A - ePWM3 output A (O)
		10	Reserved
		11	Reserved
7-6	GPIO3		Configure the GPIO3 pin as:
		00	GPIO3 - General purpose I/O 3 (default) (I/O)
		01	EPWM2B - ePWM2 output B (O)
		10	ECAP5 - eCAP5 (I/O)
		11	Reserved
5-4	GPIO2		Configure the GPIO2 pin as:
		00	GPIO2 (I/O) General purpose I/O 2 (default) (I/O)
		01	EPWM2A - ePWM2 output A (O)
		10	Reserved
		11	Reserved.
3-2	GPIO1		Configure the GPIO1 pin as:
		00	GPIO1 - General purpose I/O 1 (default) (I/O)
		01	EPWM1B - ePWM1 output B (O)
		10	ECAP - eCAP6 (I/O)
		11	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-50. GPIO Port A Multiplexing 1 (GPAMUX1) Register Field Descriptions (continued)

Bits	Field	Value	Description [1]
1-0	GPIO0		Configure the GPIO0 pin as:
		00	GPIO0 - General purpose I/O 0 (default) (I/O)
		01	EPWM1A - ePWM1 output A (O)
		10	Reserved
		11	Reserved

4.2.7.2 GPIO Port A MUX 2 (GPAMUX2) Register

The GPIO Port A MUX 2 (GPAMUX2) register is shown and described in the figure and table below.

Figure 4-43. GPIO Port A MUX 2 (GPAMUX2) Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO31		GPIO30		GPIO29		GPIO28		GPIO27		GPIO26		GPIO25		GPIO24	
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO23		GPIO22		GPIO21		GPIO20		GPIO19		GPIO18		GPIO17		GPIO16	
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-51. GPIO Port A MUX 2 (GPAMUX2) Register Field Descriptions

Bits	Field	Value	Description [1]
31-30	GPIO31		Configure the GPIO31 pin as:
		00	GPIO31 - General purpose I/O 31 (default) (I/O)
		01	Reserved
		10	Reserved
		11	EPWM9B - ePWM9 output B (O)
29-28	GPIO30		Configure the GPIO30 pin as:
		00	GPIO30 (I/O) General purpose I/O 30 (default) (I/O)
		01	Reserved
		10	Reserved
		11	EPWM9A - ePWM9 output A (O)
27-26	GPIO29		Configure the GPIO29 pin as:
		00	GPIO29 (I/O) General purpose I/O 29 (default) (I/O)
		01	SCITXDA - SCI-A transmit. (O)
		10	Reserved
		11	Reserved
25-24	GPIO28		Configure the GPIO28 pin as:
		00	GPIO28 (I/O) General purpose I/O 28 (default) (I/O)
		01	SCIRXDA - SCI-A receive (I)
		10	Reserved
		11	Reserved
23-22	GPIO27		Configure the GPIO27 pin as:
		00	GPIO27 - General purpose I/O 27 (default) (I/O)
		01	ECAP4 - eCAP4 (I/O)
		10	EQEP2S - eQEP2 strobe (I/O)
		11	Reserved

(1) If reserved configurations are selected, then the state of the pin will be undefined and the pin may be driven. These selections are reserved for future expansion and should not be used.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-51. GPIO Port A MUX 2 (GPAMUX2) Register Field Descriptions (continued)

Bits	Field	Value	Description [(1)]
21-20	GPIO26		Configure the GPIO26 pin as:
		00	GPIO26 - General purpose I/O 26 (default) (I/O)
		01	ECAP3 - eCAP3 (I/O)
		10	EQEP2I - eQEP2 index (I/O)
		11	Reserved
19-18	GPIO25		Configure the GPIO25 pin as:
		00	GPIO25 - General purpose I/O 25 (default) (I/O)
		01	ECAP2 - eCAP2 (I/O)
		10	EQEP2B - eQEP2 input B (I)
		11	Reserved
17-16	GPIO24		Configure the GPIO24 pin as:
		00	GPIO24 - General purpose I/O 24 (default) (I/O)
		01	ECAP1 - eCAP1 (I/O)
		10	EQEP2A - eQEP2 input A (I)
		11	Reserved
15-14	GPIO23		Configure the GPIO23 pin as:
		00	GPIO23 - General purpose I/O 23 (default) (I/O)
		01	EQEP1I - eQEP1 index (I/O)
		10	MFSXA - McBSP - A transmit frame synch (I/O)
		11	Reserved
13-12	GPIO22		Configure the GPIO22 pin as:
		00	GPIO22 - General purpose I/O 22 (default) (I/O)
		01	EQEP1S - eQEP1 strobe (I/O)
		10	MCLKXA - McBSP - A transmit clock (I/O)
		11	Reserved
11-10	GPIO21		Configure the GPIO21 pin as:
		00	GPIO21 - General purpose I/O 21 (default) (I/O)
		01	EQEP1B - eQEP1 input B (I)
		10	MDRA - McBSP - A data receive (I)
		11	Reserved
9-8	GPIO20		Configure the GPIO20 pin as:
		00	GPIO20 - General purpose I/O 22 (default) (I/O)
		01	EQEP1A - eQEP1 input A (I)
		10	MDXA - McBSP - A data transmit (O)
		11	Reserved
7-6	GPIO19/XCLKIN		Configure the GPIO19 pin as:
		00	GPIO19 - General purpose I/O 19 (default) (I/O)
		01	$\overline{\text{SPISTEA}}$ - SPI-A slave transmit enable (I/O)
		10	Reserved
		11	Reserved
5-4	GPIO18		Configure the GPIO18 pin as:
		00	GPIO18 - General purpose I/O 18 (default) (I/O)
		01	SPICLKA - SPI-A clock (I/O)
		10	Reserved
		11	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-51. GPIO Port A MUX 2 (GPAMUX2) Register Field Descriptions (continued)

Bits	Field	Value	Description [(1)]
3-2	GPIO17		Configure the GPIO17 pin as:
		00	GPIO17 - General purpose I/O 17 (default) (I/O)
		01	SPISOMIA - SPI-A Slave output/Master input (I/O)
		10	Reserved
		11	Reserved
1-0	GPIO16		Configure the GPIO16 pin as:
		00	GPIO16 - General purpose I/O 16 (default) (I/O)
		01	SPISIMOA - SPI-A slave-in, master-out (I/O),
		10	Reserved
		11	Reserved

4.2.7.3 GPIO Port B MUX 1 (GPBMUX1) Register

The GPIO Port B MUX 1 (GPBMUX1) register is shown and described in the figure and table below.

Figure 4-44. GPIO Port B MUX 1 (GPBMUX1) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 2**	**1 0**
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-52. GPIO Port B MUX 1 (GPBMUX1) Register Field Descriptions

Bit	Field	Value	Description
31:30	GPIO47		Configure this pin as:
		00	GPIO 47 - general purpose I/O 47 (default)
		01	Reserved
		10	Reserved
		11	Reserved
29 - 28	GPIO45		Configure this pin as:
		00	GPIO 46 - general purpose I/O 46 (default)
		01	Reserved
		10	Reserved
		11	Reserved
27 - 26	GPIO45		Configure this pin as:
		00	GPIO 45 - general purpose I/O 45 (default)
		01	Reserved
		10	Reserved
		11	Reserved
25:24	GPIO44		Configure this pin as:
		00	GPIO 44 - general purpose I/O 44 (default)
		01	Reserved
		10	Reserved
		11	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-52. GPIO Port B MUX 1 (GPBMUX1) Register Field Descriptions (continued)

Bit	Field	Value	Description
23:22	GPIO43		Configure this pin as:
		00	GPIO 43 - general purpose I/O 43 (default)
		01	Reserved
		10	Reserved
		11	Reserved
21:20	GPIO42		Configure this pin as:
		00	GPIO 42 - general purpose I/O 42 (default)
		01	Reserved
		10	Reserved
		11	Reserved
19:18	GPIO41		Configure this pin as:
		00	GPIO 41 - general purpose I/O 41 (default)
		01	Reserved
		10	Reserved
		11	Reserved
17:16	GPIO40		Configure this pin as:
		00	GPIO 40 - general purpose I/O 40 (default)
		01	Reserved
		10	Reserved
		11	Reserved
15:14	GPIO39		Configure this pin as:
		00	GPIO 39 - general purpose I/O 39 (default)
		01	Reserved
		10	Reserved
		11	Reserved
13:12	GPIO38		Configure this pin as:
		00	GPIO 38 - general purpose I/O 38 (default)
		01	Reserved
		10	Reserved
		11	Reserved
11:10	GPIO37		Configure this pin as:
		00	GPIO 37 - general purpose I/O 37 (default)
		01	ECAP2- eCAP2 (I/O)
		10	Reserved
		11	Reserved
9:8	GPIO36		Configure this pin as:
		00	GPIO 36 - general purpose I/O 36 (default)
		01	SCIRXDA - SCI - A receive data (I)
		10	Reserved
		11	Reserved
7:6	GPIO35		Configure this pin as:
		00	GPIO 35 - general purpose I/O 35 (default)
		01	SCITXDA - SCI - A transmit data (O)
		10	Reserved
		11	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-52. GPIO Port B MUX 1 (GPBMUX1) Register Field Descriptions (continued)

Bit	Field	Value	Description
5:4	GPIO34		Configure this pin as:
		00	GPIO 34 - general purpose I/O 34 (default)
		01	ECAP1 - eCAP1 (I/O)
		10	SCIRXDA - SCI - A receive data (I)
		11	XCLKOUT
3:2	GPIO33		Configure this pin as:
		00	GPIO 33 - general purpose I/O 33 (default)
		01	SCLA - I^2C clock bidirectional port (I/O)(1)
		10	EPWMSYNCO - External ePWM sync pulse output (O)
		11	$\overline{\text{ADCSOCBO}}$ - ADC start-of-conversion B (O)
1:0	GPIO32		Configure this pin as:
		00	GPIO 32 - general purpose I/O 32 (default)
		01	SDAA - I^2C data bidirectional port (I/O)(1)
		10	SCIRXDA - SCI - A receive data (I)
		11	$\overline{\text{ADCSOCAO}}$ - ADC start-of-conversion A (O)

(1) Use the M3 GPIO mux register, GPIOODR to enable the open-drain functionality of this pin for proper functionality.

4.2.7.4 GPIO Port B MUX 2 (GPBMUX2) Register

The GPIO Port B MUX 2 (GPBMUX2) register is shown and described in the figure and table below.

Figure 4-45. GPIO Port B MUX 2 (GPBMUX2) Register

31	28	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO63/XCLKIN		GPIO62		GPIO61		GPIO60		GPIO59		GPIO58		GPIO57		GPIO56
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO55		GPIO54		GPIO53		GPIO52		GPIO51		GPIO50		GPIO49		GPIO48	
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-53. GPIO Port B MUX 2 (GPBMUX2) Register Field Descriptions

Bit	Field	Value	Description
31:30	GPIO63/XCLKIN		Configure this pin as:
		00	GPIO 63 - general purpose I/O 63 (default) / XCLKIN
		01	Reserved
		10	Reserved
		11	EPWM9B - ePWM9 output B (O)
29 - 28	GPIO62		Configure this pin as:
		00	GPIO 62 - general purpose I/O 62 (default)
		01	Reserved
		10	Reserved
		11	EPWM9A - ePWM9 output A (O)
27 - 26	GPIO61		Configure this pin as:
		00	GPIO 61 - general purpose I/O 61 (default)
		01	Reserved
		10	Reserved
		11	EPWM8B - ePWM8 output B (O)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-53. GPIO Port B MUX 2 (GPBMUX2) Register Field Descriptions (continued)

Bit	Field	Value	Description
25:24	GPIO60		Configure this pin as:
		00	GPIO 60 - general purpose I/O 60 (default)
		01	Reserved
		10	Reserved
		11	EPWM8A - ePWM8 output A (O)
23:22	GPIO59		Configure this pin as:
		00	GPIO 59 - general purpose I/O 59 (default)
		01	MFSRA - McBSP - A receive frame synch (I/O)
		10	Reserved
		11	EPWM7B - ePWM7 output B (O)
21:20	GPIO58		Configure this pin as:
		00	GPIO 58 - general purpose I/O 58 (default)
		01	MCLKRA - McBSP - A receive clock
		10	Reserved
		11	EPWM7A - ePWM7 output A (O)
19:18	GPIO57		Configure this pin as:
		00	GPIO 571 - general purpose I/O 57 (default)
		01	$\overline{\text{SPISTEA}}$ - SPI - A slave transmit enable (I/O)
		10	Reserved
		11	EQEP3I - eQEP3 index (I/O)
17:16	GPIO56		Configure this pin as:
		00	GPIO 56 - general purpose I/O 56 (default)
		01	SPICLKA - SPI - A clock (I/O)
		10	Reserved
		11	EQEP3S - eQEP3 strobe (/O)
15:14	GPIO55		Configure this pin as:
		00	GPIO 55 - general purpose I/O 55 (default)
		01	SPISOMIA - SPI - A slave out, master in (I/O)
		10	Reserved
		11	EQEP3B - eQEP3 input B (I)
13:12	GPIO54		Configure this pin as:
		00	GPIO 54 - general purpose I/O 54 (default)
		01	SPISIMOA - SPI - A slave in, master out (I/O)
		10	Reserved
		11	EQEP3A - eQEP3 input A (I)
11:10	GPIO53		Configure this pin as:
		00	GPIO 53 - general purpose I/O 53 (default)
		01	EQEP1I - eQEP1 index (I/O)
		10	Reserved
		11	Reserved
9:8	GPIO52		Configure this pin as:
		00	GPIO 52 - general purpose I/O 52 (default)
		01	EQEP1S - eQEP1 strobe (I/O)
		10	Reserved
		11	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-53. GPIO Port B MUX 2 (GPBMUX2) Register Field Descriptions (continued)

Bit	Field	Value	Description
7:6	GPIO51		Configure this pin as:
		00	GPIO 51 - general purpose I/O 51 (default)
		01	EQEP1B - eQEP1 input B (I)
		10	Reserved
		11	Reserved
5:4	GPIO50		Configure this pin as:
		00	GPIO 50 - general purpose I/O 50 (default)
		01	EQEP1A- eQEP1 input A (I)
		10	Reserved
		11	Reserved
3:2	GPIO49		Configure this pin as:
		00	GPIO 49 - general purpose I/O 49 (default)
		01	ECAP6 - eCAP6 (I/O)
		10	Reserved
		11	Reserved
1:0	GPIO48		Configure this pin as:
		00	GPIO 48 - general purpose I/O 48 (default)
		01	ECAP5 - eCAP5 (I/O)
		10	Reserved
		11	Reserved

4.2.7.5 GPIO Port C MUX 1 (GPCMUX1) Register

The GPIO Port C MUX 1 (GPCMUX1) register is shown and described in the figure and table below.

Figure 4-46. GPIO Port C MUX 1 (GPCMUX1) Register

31										16
Reserved										
R-0										
15	14	13	12	11	10	9	8	7		0
GPIO71		GPIO70		GPIO69		GPIO68		Reserved		
R/W-0		R/W-0		R/W-0		R/W-0		R-0		

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-54. GPIO Port C MUX 1 (GPCMUX1) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-14	GPIO71		Configure this pin as:
		00	GPIO 71 - general purpose I/O 71 (default)
		01	Reserved
		10	Reserved
		11	Reserved
13-12	GPIO70		Configure this pin as:
		00	GPIO 70 - general purpose I/O 70 (default)
		01	Reserved
		10	Reserved
		11	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-54. GPIO Port C MUX 1 (GPCMUX1) Register Field Descriptions (continued)

Bit	Field	Value	Description
11-10	GPIO69		Configure this pin as:
		00	GPIO 69 - general purpose I/O 69 (default)
		01	Reserved
		10	Reserved
		11	Reserved
9-8	GPIO68		Configure this pin as:
		00	GPIO 68 - general purpose I/O 68 (default)
		01	Reserved
		10	Reserved
		11	Reserved
7-0	Reserved		Reserved

4.2.7.6 GPIO Port E MUX 1 (GPEMUX1) Register

The GPIO Port E MUX 1 (GPEMUX1) register is shown and described in the figure and table below.

Figure 4-47. GPIO Port E MUX 1 (GPEMUX1) Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO135		GPIO134		GPIO133		GPIO132		GPIO131		GPIO130		GPIO129		GPIO128	
R/W-0		R/W-0		R/W-0		R/W-0		R/W0		R/W0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-55. GPIO Port E MUX 1 (GPEMUX1) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-14	GPIO135		Configure this pin as:
		00	GPIO 135 - general purpose I/O 135 (default)
		01	Reserved
		10	Reserved
		11	COMP5OUT - Comparator 5 output (O)
13-12	GPIO134		Configure this pin as:
		00	GPIO 134 - general purpose I/O 134 (default)
		01	Reserved
		10	Reserved
		11	Reserved
11-10	GPIO133		Configure this pin as:
		00	GPIO 133 - general purpose I/O 133 (default)
		01	Reserved
		10	Reserved
		11	COMP4OUT - Comparator 4 output (O)
9-8	GPIO132		Configure this pin as:
		00	GPIO 132 - general purpose I/O 132 (default)
		01	Reserved
		10	Reserved
		11	COMP3OUT - Comparator 3 output (O)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-55. GPIO Port E MUX 1 (GPEMUX1) Register Field Descriptions (continued)

Bit	Field	Value	Description
7-6	GPIO131		Configure this pin as:
		00	GPIO 131 - general purpose I/O 131 (default)
		01	Reserved
		10	Reserved
		11	COMP2OUT - Comparator 2 output (O)
5-4	GPIO130		Configure this pin as:
		00	GPIO 130 - general purpose I/O 130 (default)
		01	Reserved
		10	Reserved
		11	COMP6OUT - Comparator 6 output (O)
3-2	GPIO129		Configure this pin as:
		00	GPIO 129 - general purpose I/O 129 (default)
		01	Reserved
		10	Reserved
		11	COMP1OUT - Comparator 1 output (O)
1-0	GPIO128		Configure this pin as:
		00	GPIO 128 - general purpose I/O 128 (default)
		01	Reserved
		10	Reserved
		11	Reserved

4.2.7.7 Analog I/O MUX 1 (AIOMUX1) Register

The Analog I/O MUX 1 (AIOMUX1) register is shown and described in the figure and table below.

Figure 4-48. Analog I/O MUX 1 (AIOMUX1) Register

31	30	29	28	27	26	25	24	23	22	21	20	19	16
Reserved		Reserved		Reserved		AI012		Reserved		Reserved		Reserved	
R/W-0x2		R/W-0x2		R/W-0x2		R/W-0x2		R/W-0x2		R/W-0x2		R/W-0xA	

15	14	13	12	11	10	9	8	7	6	5	4	3	0
Reserved		AIO6		Reserved		AIO4		Reserved		AIO2		Reserved	
R/W-0x2		R/W-0x2		R/W-0x2		R/W-0x2		R/W-0x2		R/W-0x2		R/W-0xA	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-56. Analog I/O MUX 1 (AIOMUX1) Register Field Descriptions

Bit	Field	Value	Description
31:26	Reserved		Any writes to these bit(s) must always have a value of 0x2A
25:24	AIO12	00 or 01	AIO12 enabled
		10 or 11	AIO12 disabled (default)
23:20	Reserved		Any writes to these bit(s) must always have a value of 0xA
19:16	Reserved		Any writes to these bit(s) must always have a value of 0xA
15-14	Reserved		Any writes to these bit(s) must always have a value of 0x2
13:12	AIO6	00 or 01	AIO6 enabled
		10 or 11	AIO6 disabled (default)
11:10	Reserved		Any writes to these bit(s) must always have a value of 0x2
9:8	AIO4	00 or 01	AIO4 enabled
		10 or 11	AIO4 disabled (default)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-56. Analog I/O MUX 1 (AIOMUX1) Register Field Descriptions (continued)

Bit	Field	Value	Description
7:6	Reserved		Any writes to these bit(s) must always have a value of 0x2
5:4	AIO2	00 or 01 10 or 11	AIO2 enabled AIO2 disabled (default)
3:0	Reserved		Any writes to these bit(s) must always have a value of 0xA

4.2.7.8 Analog I/O MUX 2 (AIOMUX2) Register

The Analog I/O MUX 2 (AIOMUX2) register is shown and described in the figure and table below.

Figure 4-49. Analog I/O MUX 2 (AIOMUX2) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 16
Reserved	Reserved	Reserved	AIO28	Reserved	Reserved	Reserved
R/W-0x2	R/W-0x2	R/W-0x2	R/W-0x2	R/W-0x2	R/W-0x2	R/W-0xA
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 0**
Reserved	AIO22	Reserved	AIO20	Reserved	AIO18	Reserved
R/W-0x2	R/W-0x2	R/W-0x2	R/W-0x2	R/W-0x2	R/W-0x2	R/W-0xA

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-57. Analog I/O MUX 2 (AIOMUX2) Register Field Descriptions

Bit	Field	Value	Description
31:26	Reserved		Any writes to these bit(s) must always have a value of 0x2A
25:24	AIO28	00 or 01 10 or 11	AIO28 enabled AIO28 disabled (default)
23:20	Reserved		Any writes to these bit(s) must always have a value of 0xA
19:16	Reserved		Any writes to these bit(s) must always have a value of 0xA
15-14	Reserved		Any writes to these bit(s) must always have a value of 0x2
13:12	AIO22	00 or 01 10 or 11	AIO22 enabled AIO22 disabled (default)
11:10	Reserved		Any writes to these bit(s) must always have a value of 0x2
9:8	AIO20	00 or 01 10 or 11	AIO20 enabled AIO20 disabled (default)
7:6	Reserved		Any writes to these bit(s) must always have a value of 0x2
5:4	AIO18	00 or 01 10 or 11	AIO18 enabled AIO18 disabled (default)
3:0	Reserved		Any writes to these bit(s) must always have a value of 0xA

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.9 GPIO Port A Qualification Control (GPACTRL) Register

The GPIO Port A Qualification Control (GPACTRL) register is shown and described in the figure and table below.

Figure 4-50. GPIO Port A Qualification Control (GPACTRL) Register

31	24	23	16
QUALPRD3		QUALPRD2	
R/W-0		R/W-0	
15	8	7	0
QUALPRD1		QUALPRD0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

The GPxCTRL registers specify the sampling period for input pins when configured for input qualification using a window of 3 or 6 samples. The sampling period is the amount of time between qualification samples relative to the period of SYSCLKOUT. The number of samples is specified in the GPxQSELn registers.

Table 4-58. GPIO Port A Qualification Control (GPACTRL) Register Field Descriptions

Bits	Field	Value	Description (1)
31-24	QUALPRD3		Specifies the sampling period for pins GPIO24 to GPIO31.
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
23-16	QUALPRD2		Specifies the sampling period for pins GPIO16 to GPIO23.
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
15-8	QUALPRD1		Specifies the sampling period for pins GPIO8 to GPIO15.
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
7-0	QUALPRD0		Specifies the sampling period for pins GPIO0 to GPIO7.
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$

(1) This register is EALLOW protected.

(2) $T_{SYSCLKOUT}$ indicates the period of SYSCLKOUT.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.10 GPIO Port B Qualification Control (GPBCTRL) Register

The GPIO Port B Qualification Control (GPBCTRL) register is shown and described in the figure and table below.

Figure 4-51. GPIO Port B Qualification Control (GPBCTRL) Register

31	24	23	16
QUALPRD3		QUALPRD2	
R/W-0		R/W-0	
15	8	7	0
QUALPRD1		QUALPRD0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-59. GPIO Port B Qualification Control (GPBCTRL) Register Field Descriptions

Bits	Field	Value	Description [1]
31-24	QUALPRD3		Specifies the sampling period for pins GPIO56 to GPIO63
		0x00	Sampling Period = $T_{SYSCLKOUT}$ [2]
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
23-16	QUALPRD2		Specifies the sampling period for pins GPIO48 to GPIO55
		0x00	Sampling Period = $T_{SYSCLKOUT}$ [2]
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
15-8	QUALPRD1		Specifies the sampling period for pins GPIO40 to GPIO47
		0x00	Sampling Period = $T_{SYSCLKOUT}$ [2]
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
7-0	QUALPRD0		Specifies the sampling period for pins GPIO32 to GPIO 39
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
		0x00	Sampling Period = $T_{SYSCLKOUT}$ [2]
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$

(1) This register is EALLOW protected.
(2) $T_{SYSCLKOUT}$ indicates the period of SYSCLKOUT.

4.2.7.11 GPIO Port C Qualification Control (GPCCTRL) Register

The GPIO Port C Qualification Control (GPCCTRL) register is shown and described in the figure and table below.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-52. GPIO Port C Qualification Control (GPCCTRL) Register

31	8	7	0
Reserved		QUALPRDO	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-60. GPIO Port C Qualification Control (GPCCTRL) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-0	QUALPRDO		Specifies the sampling period for pins GPIO68 TO GPIO71
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (1)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$

(1) $T_{SYSCLKOUT}$ indicates the period of SYSCLKOUT.

4.2.7.12 GPIO Port E Qualification Control (GPECTRL) Register

The GPIO Port E Qualification Control (GPECTRL) register is shown and described in the figure and table below.

Figure 4-53. GPIO Port E Qualification Control (GPECTRL) Register

31	8	7	0
Reserved		QUALPRDO	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-61. GPIO Port E Qualification Control (GPECTRL) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-0	QUALPRDO		Specifies the sampling period for pins GPIO128 TO GPIO135
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (1)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
			Note: GPIO on Port E is synchronized to the analog subsystem clock by default.

(1) $T_{SYSCLKOUT}$ indicates the period of SYSCLKOUT.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.13 GPIO Port A Qualification Select 1 (GPAQSEL1) Register

The GPIO Port A Qualification Select 1 (GPAQSEL1) register is shown and described in the figure and table below.

Figure 4-54. GPIO Port A Qualification Select 1 (GPAQSEL1) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-62. GPIO Port A Qualification Select 1 (GPAQSEL1) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO15-GPIO0		Select input qualification type for GPIO0 to GPIO15. The input qualification of each GPIO input is controlled by two bits .
		00	Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		11	Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.

(1) This register is EALLOW protected.

4.2.7.14 GPIO Port A Qualification Select 2 (GPAQSEL2) Register

The GPIO Port A Qualification Select 2 (GPAQSEL2) register is shown and described in the figure and table below.

Figure 4-55. GPIO Port A Qualification Select 2 (GPAQSEL2) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-63. GPIO Port A Qualification Select 2 (GPAQSEL2) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO31-GPIO16		Select input qualification type for GPIO16 to GPIO31. The input qualification of each GPIO input is controlled by two bits.
		00	Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		11	Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.

(1) This register is EALLOW protected.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.15 GPIO Port B Qualification Select 1 (GPBQSEL1) Register

The GPIO Port B Qualification Select 1 (GPBQSEL1) register is shown and described in the figure and table below.

Figure 4-56. GPIO Port B Qualification Select 1 (GPBQSEL1) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-64. GPIO Port B Qualification Select 1 (GPBQSEL1) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO47-GPIO32		Select input qualification type for GPIO32 to GPIO47 . The input qualification of each GPIO input is controlled by two bits.
		00	Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPBCTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPBCTRL register.
		11	Asynchronous (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.

(1) This register is EALLOW protected.

4.2.7.16 GPIO Port B Qualification Select 2 (GPBQSEL2) Register

The GPIO Port B Qualification Select 2 (GPBQSEL2) register is shown and described in the figure and table below.

Figure 4-57. GPIO Port B Qualification Select 2 (GPBQSEL2) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-65. GPIO Port B Qualification Select 2 (GPBQSEL2) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO63-GPIO48		Select input qualification type for GPIO48 to GPIO63 . The input qualification of each GPIO input is controlled by two bits.
		00	Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPBCTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPBCTRL register.
		11	Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.

(1) This register is EALLOW protected.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.17 GPIO Port C Qualification Select 1 (GPCQSEL1) Register

The GPIO Port C Qualification Select 1 (GPCQSEL1) register is shown and described in the figure and table below.

Figure 4-58. GPIO Port C Qualification Select 1 (GPCQSEL1) Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO71		GPIO70		GPIO69		GPIO68		Reserved							
R/W-0		R/W-0		R/W-0		R/W-0		R-0							

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-66. GPIO Port C Qualification Select 1 (GPCQSEL1) Register Field Descriptions

Bits	Field	Value	Description [1]
31-16	Reserved		Any writes to these bit(s) must always have a value of 0.
15-8	GPIO71-GPIO68		Select input qualification type for GPIO68 to GPIO71 . The input qualification of each GPIO input is controlled by two bits.
		00	Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPCCTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPCCTRL register.
		11	Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.
7-0			Any writes to these bit(s) must always have a value of 0.

(1) This register is EALLOW protected.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.18 GPIO Port E Qualification Select 1 (GPEQSEL1) Register

The GPIO Port E Qualification Select 1 (GPEQSEL1) register is shown and described in the figure and table below.

Figure 4-59. GPIO Port E Qualification Select 1 (GPEQSEL1) Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
GPIO135		GPIO134		GPIO133		GPIO132		GPIO131		GPIO130		GPIO129		GPIO128	
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-67. GPIO Port E Qualification Select 1 (GPEQSEL1) Register Field Descriptions

Bits	Field	Value	Description [(1)]
31-16	Reserved		Any writes to these bit(s) must always have a value of 0.
15-0	GPIO135-GPIO128		Select input qualification type for GPIO128 to GPIO135 . The input qualification of each GPIO input is controlled by two bits.
		00	Synchronize to analog subsystem clock. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPECTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPECTRL register.
		11	Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to analog subsystem clock.

(1) This register is EALLOW protected.

The GPxDIR registers control the direction of the pins when they are configured as a GPIO in the appropriate MUX register. The direction register has no effect on pins configured as peripheral functions.

4.2.7.19 GPIO Port A Direction (GPADIR) Register

The GPIO Port A Direction (GPADIR) register is shown and described in the figure and table below.

Figure 4-60. GPIO Port A Direction (GPADIR) Register

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-68. GPIO Port A Direction (GPADIR) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO31-GPIO0		Controls direction of GPIO Port A pins when the specified pin is configured as a GPIO in the appropriate GPAMUX1 or GPAMUX2 register.
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output
			The value currently in the GPADAT output latch is driven on the pin. To initialize the GPADAT latch prior to changing the pin from an input to an output, use the GPASET, GPACLEAR, and GPATOGGLE registers.

(1) This register is EALLOW protected.

4.2.7.20 GPIO Port B Direction (GPBDIR) Register

The GPIO Port B Direction (GPBDIR) register is shown and described in the figure and table below.

Figure 4-61. GPIO Port B Direction (GPBDIR) Register

31	30	29	28	27	26	25	24
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-69. GPIO Port B Direction (GPBDIR) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO63-GPIO32		Controls direction of GPIO pin when GPIO mode is selected. Reading the register returns the current value of the register setting
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output

(1) This register is EALLOW protected.

4.2.7.21 GPIO Port C Direction (GPCDIR) Register

The GPIO Port C Direction (GPCDIR) register is shown and described in the figure and table below.

Figure 4-62. GPIO Port C Direction (GPCDIR) Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								GPIO71	GPIO70	GPIO69	GPIO68	Reserved			
R-0								R/W-0	R/W-0	R/W-0	R/W-0	R-0			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 4-70. GPIO Port C Direction (GPCDIR) Register Field Descriptions

Bits	Field	Value	Description [1]
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-4	GPIO71-GPIO68		Controls direction of GPIO pin when GPIO mode is selected. Reading the register returns the current value of the register setting
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output
3-0	Reserved		Any writes to these bit(s) must always have a value of 0.

(1) This register is EALLOW protected.

4.2.7.22 GPIO Port E Direction (GPEDIR) Register

The GPIO Port E Direction (GPEDIR) register is shown and described in the figure and table below.

Figure 4-63. GPIO Port E Direction (GPEDIR) Register

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

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-71. GPIO Port E Direction (GPEDIR) Register Field Descriptions

Bits	Field	Value	Description [1]
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-0	GPIO135-GPIO128		Controls direction of GPIO pin when GPIO mode is selected. Reading the register returns the current value of the register setting
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output

(1) This register is EALLOW protected.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.23 GPIO Port E Pullup Disable (GPEPUD) Register

The GPIO Port E Pullup Disable (GPEPUD) register is shown and described in the figure and table below.

Figure 4-64. GPIO Port E Pullup Disable (GPEPUD)

31	30	29	28	27	26	25	24
Reserved							
R/W-0							
23	22	21	20	19	18	17	16
Reserved							
R/W-0							
15	14	13	12	11	10	9	8
Reserved							
R/W-0xF							
7	6	5	4	3	2	1	0
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-72. GPIO Port E Pullup Disable (GPEPUD) Register Field Descriptions

Bits	Field	Value	Description [(1)]
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-0	GPIO135-GPIO128		Configures the internal pullup resistor on the selected GPIO Port E pin. Each GPIO pin corresponds to one bit in this register.
		0	Enable the internal pullup on the specified pin.
		1	Disable the intenral pullup on the specified pin (default).

(1) All other pins' pullup functionality is controlled by the GPIOPUR register located in the M3 GPIO mux register rspace.

4.2.7.24 Analog I/O DIR (AIODIR) Register

The Analog I/O DIR (AIODIR) register is shown and described in the figure and table below.

Figure 4-65. Analog I/O DIR (AIODIR) Register

31	30	29	28	27	26	25	24
Reserved	AIO30	Reserved	AIO28	Reserved	AIO26	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	
23	22	21	20	19	18	17	16
Reserved	AIO22	Reserved	AIO20	Reserved	AIO18	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	
15	14	13	12	11	10	9	8
Reserved	AIO14	Reserved	AIO12	Reserved	AIO10	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	
7	6	5	4	3	2	1	0
Reserved	AIO6	Reserved	AIO4	Reserved	AIO2	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after resetR/W-x

Table 4-73. Analog I/O DIR (AIODIR) Register Field Descriptions

Bit	Field	Value	Description
31:0	AIOn		Controls direction of the avaliable AIO pin when AIO mode is selected. Reading the register returns the current value of the register setting.
		0	Configures the AIO pin as an input. (default)
		1	Configures the AIO pin as an output

Copyright © 2012–2019, Texas Instruments Incorporated

The GPIO data registers indicate the current status of the GPIO pin, irrespective of which mode the pin is in. Writing to this register will set the respective GPIO pin high or low if the pin is enabled as a GPIO output, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A reset will clear all bits and latched values to zero. The value read from the GPxDAT registers reflect the state of the pin (after qualification), not the state of the output latch of the GPxDAT register.

Typically the DAT registers are used for reading the current state of the pins. To easily modify the output level of the pin refer to the SET, CLEAR and TOGGLE registers.

4.2.7.25 GPIO Port A Data (GPADAT) Register

The GPIO Port A Data (GPADAT) register is shown and described in the figure and table below.

Figure 4-66. GPIO Port A Data (GPADAT) Register

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset[(1)]

(1) x = The state of the GPADAT register is unknown after reset. It depends on the level of the pin after reset.

Table 4-74. GPIO Port A Data (GPADAT) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31-GPIO0		Each bit corresponds to one GPIO port A pin (GPIO0-GPIO31)
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode for which the pin is configured.
			Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPAMUX1/2 and GPADIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode for which the pin is configured.
			Writing a 1will force an output of 1if the pin is configured as a GPIO output in the appropriate GPAMUX1/2 and GPADIR registers; otherwise, the value is latched but not used to drive the pin.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.26 GPIO Port B Data (GPBDAT) Register

The GPIO Port B Data (GPBDAT) register is shown and described in the figure and table below.

Figure 4-67. GPIO Port B Data (GPBDAT) Register

31	30	29	28	27	26	25	24
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
23	22	21	20	19	18	17	16
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
15	14	13	12	11	10	9	8
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
7	6	5	4	3	2	1	0
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-75. GPIO Port B Data (GPBDAT) Register Field Descriptions

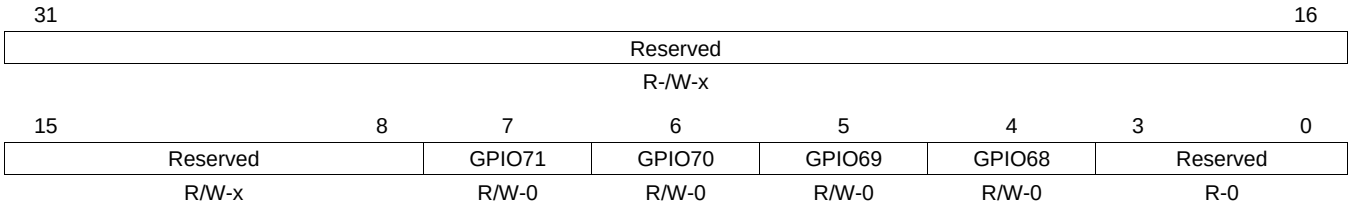
Bit	Field	Value	Description
31-0	GPIO63-GPIO32		Each bit corresponds to one GPIO port B pin (GPIO32-GPIO63)
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode for which the pin is configured..
			Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPBMUX1/2 and GPBDIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode for which the pin is configured.
			Writing a 1 will force an output of 1 if the pin is configured as a GPIO output in the GPBMUX1 and GPBDIR registers; otherwise, the value is latched but not used to drive the pin.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.27 GPIO Port C Data (GPCDAT) Register

The GPIO Port C Data (GPCDAT) register is shown and described in the figure and table below.

Figure 4-68. GPIO Port C Data (GPCDAT) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-76. GPIO Port C Data (GPCDAT) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-4	GPIO71-GPIO68		Each bit corresponds to one GPIO port C pin (GPIO68-GPIO71)
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode for which the pin is configured.
			Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPCMUX1/2 and GPCDIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode for which the pin is configured.
			Writing a 1 will force an output of 1 if the pin is configured as a GPIO output in the GPCMUX1 and GPCDIR registers; otherwise, the value is latched but not used to drive the pin.
3-0	Reserved		Any writes to these bit(s) must always have a value of 0.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.28 GPIO Port E Data (GPEDAT) Register

The GPIO Port E Data (GPEDAT) register is shown and described in the figure and table below.

Figure 4-69. GPIO Port E Data (GPEDAT) Register

31							16
Resrerved							
R-0							
15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-77. GPIO Port E Data (GPEDAT) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-0	GPIO135-GPIO128		Each bit corresponds to one GPIO port E pin (GPIO128-GPIO135)
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode the pin is configured for. Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPEMUX1 and GPEDIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode the pin is configured for. Writing a 1 will force an output of 1 if the pin is configured as a GPIO output in the GPEMUX1 and GPEDIR registers; otherwise, the value is latched but not used to drive the pin.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.29 Analog I/O DAT (AIODAT) Register

The Analog I/O DAT (AIODAT) register is shown and described in the figure and table below.

Figure 4-70. Analog I/O DAT (AIODAT) Register

31	30	29	28	27	26	25	24
Reserved	AIO30	Reserved	AIO28	Reserved	AIO26	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	
23	22	21	20	19	18	17	16
Reserved	AIO22	Reserved	AIO20	Reserved	AIO18	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	
15	14	13	12	11	10	9	8
Reserved	AIO14	Reserved	AIO12	Reserved	AIO10	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	
7	6	5	4	3	2	1	0
Reserved	AIO6	Reserved	AIO4	Reserved	AIO2	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after resetR/W-x

Table 4-78. Analog I/O DAT (AIODAT) Register Field Descriptions

Bit	Field	Value	Description
31-0	AIOn		Each bit corresponds to one AIO port pin
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode for which the pin is configured. Writing a 0 will force an output of 0 if the pin is configured as a AIO output in the appropriate registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode for which the pin is configured. Writing a 1will force an output of 1if the pin is configured as a AIO output in the appropriate registers; otherwise, the value is latched but not used to drive the pin.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.30 GPIO Port A Set, Clear and Toggle (GPASET, GPACLEAR, GPATOGGLE) Registers

The GPIO Port A Set, Clear and Toggle (GPASET, GPACLEAR, GPATOGGLE) registers are shown and described in the figure and table below.

Figure 4-71. GPIO Port A Set, Clear and Toggle (GPASET, GPACLEAR, GPATOGGLE) Registers

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-79. GPIO Port A Set (GPASET) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31-GPIO0		Each GPIO port A pin (GPIO0-GPIO31) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to high. If the pin is configured as a GPIO output then it will be driven high. If the pin is not configured as a GPIO output then the latch is set high but the pin is not driven.

Table 4-80. GPIO Port A Clear (GPACLEAR) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31 - GPIO0		Each GPIO port A pin (GPIO0-GPIO31) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to low. If the pin is configured as a GPIO output then it will be driven low. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Table 4-81. GPIO Port A Toggle (GPATOGGLE) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31-GPIO0		Each GPIO port A pin (GPIO0-GPIO31) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to toggle from its current state. If the pin is configured as a GPIO output then it will be driven in the opposite direction of its current state. If the pin is not configured as a GPIO output then the latch is toggled but the pin is not driven.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.31 GPIO Port B Set, Clear and Toggle (GPBSET, GPBCLEAR, GPBTOGGLE) Registers

The GPIO Port B Set, Clear and Toggle (GPBSET, GPBCLEAR, GPBTOGGLE) registers are shown and described in the figure and table below.

Figure 4-72. GPIO Port B Set, Clear and Toggle (GPBSET, GPBCLEAR, GPBTOGGLE) Registers

31	30	29	28	27	26	25	24
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
23	22	21	20	19	18	17	16
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
15	14	13	12	11	10	9	8
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
7	6	5	4	3	2	1	0
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-82. GPIO Port B Set (GPBSET) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO63 -GPIO32		Each GPIO port B pin (GPIO32-GPIO63) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to high. If the pin is configured as a GPIO output then it will be driven high. If the pin is not configured as a GPIO output then the latch is set but the pin is not driven.

Table 4-83. GPIO Port B Clear (GPBCLEAR) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO63 -GPIO32		Each GPIO port B pin (GPIO32-GPIO63) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to low. If the pin is configured as a GPIO output then it will be driven low. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Table 4-84. GPIO Port B Toggle (GPBTOGGLE) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO63 -GPIO32		Each GPIO port B pin (GPIO32-GPIO63) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to toggle from its current state. If the pin is configured as a GPIO output then it will be driven in the opposite direction of its current state. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.32 GPIO Port C Set, Clear and Toggle (GPCSET, GPCCLEAR, GPCTOGGLE) Registers

The GPIO Port C Set, Clear and Toggle (GPCSET, GPCCLEAR, GPCTOGGLE) registers are shown and described in the figure and table below.

Figure 4-73. GPIO Port C Set, Clear and Toggle (GPCSET, GPCCLEAR, GPCTOGGLE) Registers

31					24
Reserved					
R-0					
23					16
Reserved					
R-0					
15					8
Reserved					
R-0					

7	6	5	4	3	0
GPIO71	GPIO70	GPIO69	GPIO68	Reserved	
R/W-x	R/W-x	R/W-x	R/W-x	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-85. GPIO Port C Set (GPCSET) Register Field Descriptions

Bits	Field	Value	Description
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-4	GPIO71-GPIO68		Each GPIO port C pin (GPIO68-GPIO71) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to high. If the pin is configured as a GPIO output then it will be driven high. If the pin is not configured as a GPIO output then the latch is set but the pin is not driven.
3-0	Reserved		Any writes to these bit(s) must always have a value of 0.

Table 4-86. GPIO Port C Clear (GPCCLEAR) Register Field Descriptions

Bits	Field	Value	Description
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-4	GPIO71 -GPIO68		Each GPIO port C pin (GPIO68-GPIO71) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to low. If the pin is configured as a GPIO output then it will be driven low. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.
3-0	Reserved		Any writes to these bit(s) must always have a value of 0.

Table 4-87. GPIO Port C Toggle (GPCTOGGLE) Register Field Descriptions

Bits	Field	Value	Description
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-4	GPIO71 -GPIO68		Each GPIO port C pin (GPIO68-GPIO71) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to toggle from its current state. If the pin is configured as a GPIO output then it will be driven in the opposite direction of its current state. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.
3-0	Reserved		Any writes to these bit(s) must always have a value of 0.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.33 GPIO Port E Set, Clear and Toggle (GPESET, GPECLEAR, GPETOGGLE) Registers

The GPIO Port E Set, Clear and Toggle (GPESET, GPECLEAR, GPETOGGLE) registers are shown and described in the figure and table below.

Figure 4-74. GPIO Port E Set, Clear and Toggle (GPESET, GPECLEAR, GPETOGGLE) Registers

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-88. GPIO Port E Set (GPESET) Register Field Descriptions

Bits	Field	Value	Description
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-0	GPIO135 -GPIO128		Each GPIO port E pin (GPIO128-GPIO135) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to high. If the pin is configured as a GPIO output then it will be driven high. If the pin is not configured as a GPIO output then the latch is set but the pin is not driven.

Table 4-89. GPIO Port E Clear (GPECLEAR) Register Field Descriptions

Bits	Field	Value	Description
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-0	GPIO135 -GPIO128		Each GPIO port E pin (GPIO128-GPIO135) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to low. If the pin is configured as a GPIO output then it will be driven low. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Table 4-90. GPIO Port E Toggle (GPETOGGLE) Register Field Descriptions

Bits	Field	Value	Description
31-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-0	GPIO135 -GPIO128		Each GPIO port E pin (GPIO128-GPIO135) corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to toggle from its current state. If the pin is configured as a GPIO output then it will be driven in the opposite direction of its current state. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.34 Analog I/O Toggle (AIOSET, AIOCLEAR, AIOTOGGLE) Register

The Analog I/O Toggle (AIOSET, AIOCLEAR, AIOTOGGLE) register is shown and described in the figure and table below.

Figure 4-75. Analog I/O Toggle (AIOSET, AIOCLEAR, AIOTOGGLE) Register

31	30	29	28	27	26	25	24
Reserved	AIO30	Reserved	AIO28	Reserved	AIO26	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	
23	22	21	20	19	18	17	16
Reserved	AIO22	Reserved	AIO20	Reserved	AIO18	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	
15	14	13	12	11	10	9	8
Reserved	AIO14	Reserved	AIO12	Reserved	AIO10	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	
7	6	5	4	3	2	1	0
Reserved	AIO6	Reserved	AIO4	Reserved	AIO2	Reserved	
R-0	R/W-x	R-0	R/W-x	R-0	R/W-x	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after resetR/W-x

Table 4-91. Analog I/O Set (AIOSET) Register Field Descriptions

Bits	Field	Value	Description
31-0	AIOn		Each AIO pin corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to high. If the pin is configured as a AIO output then it will be driven high. If the pin is not configured as a AIO output then the latch is set but the pin is not driven.

Table 4-92. Analog I/O Clear (AIOCLEAR) Register Field Descriptions

Bits	Field	Value	Description
31-0	AIOn		Each AIO pin corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to low. If the pin is configured as a AIO output then it will be driven low. If the pin is not configured as a AIO output then the latch is cleared but the pin is not driven.

Table 4-93. Analog I/O Toggle (AIOTOGGLE) Register Field Descriptions

Bits	Field	Value	Description
31-0	AIOn		Each AIO pin corresponds to one bit in this register.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to toggle from its current state. If the pin is configured as a AIO output then it will be driven in the opposite direction of its current state. If the pin is not configured as a AIO output then the latch is cleared but the pin is not driven.

Copyright © 2012–2019, Texas Instruments Incorporated

4.2.7.35 GPIO Trip Input Select (GPTRIPxSEL) Register

The GPIO Trip Input Select (GPTRIPxSEL) register is shown and described in the figure and table below.

Figure 4-76. GPIO Trip Input Select Register (GPTRIPxSEL)

15	6	5	0
Reserved		GPTRIPxSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-94. GPIO Trip Input Select Register (GPTRIPxSEL) Field Descriptions

Bits	Field	Value	Description [(1)]
15-6	Reserved		Reserved
5-0	GPTRIPxSEL		Select GPIO input for GPTRIP signal: .
		000000	Select the GPIO0
		000001	Select the GPIO1
		. . .	. . .
		111110	Select the GPIO62
		111111	Select the GPIO63

(1) This register is EALLOW protected.

Note: There are 12 GPTRIP select registers and each has a specific input signal associated with it. Table 4-95 shows the input signal mapping.

Table 4-95. GPTRIP Input Signals

Register	GPTRIP Input Signals
GPTRIP1SEL	TZ1n and TRIPIN1
GPTRIP2SEL	TZ2n, ADCEXTTRIG, and TRIPIN2
GPTRIP3SEL	TZ3n and TRIPIN3
GPTRIP4SEL	XINT1 and TRIPIN4
GPTRIP5SEL	XINT2 and TRIPIN5
GPTRIP6SEL	XINT3 and TRIPIN6
GPTRIP7SEL	ECAP1 and TRIPIN7
GPTRIP8SEL	ECAP2 and TRIPIN8
GPTRIP9SEL	ECAP3 and TRIPIN9
GPTRIP10SEL	ECAP4 and TRIPIN10
GPTRIP11SEL	ECAP5 and TRIPIN11
GPTRIP12SEL	ECAP6 and TRIPIN12

4.2.7.36 GPIO Low Power Mode Wakeup Select 1 (GPIOLPMSEL1) Register

The GPIO Low Power Mode Wakeup Select 1 (GPIOLPMSEL1) register is shown and described in the figure and table below.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 4-77. GPIO Low Power Mode Wakeup Select 1 (GPIOLPMSEL1) Register

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-96. GPIO Low Power Mode Wakeup Select 1 (GPIOLPMSEL1) Register Field Descriptions

Bits	Field	Value	Description [(1)]
31-0	GPIO31 - GPIO0		Low Power Mode Wakeup Selection 1. Each bit in this register corresponds to one GPIO port A pin (GPIO0 - GPIO31).
		0	If the bit is cleared, the signal on the corresponding pin will have no effect on the HALT and STANDBY low power modes.
		1	If the respective bit is set to 1, the signal on the corresponding pin is able to wake the device from both HALT and STANDBY low power modes.

(1) This register is EALLOW protected.

4.2.7.37 GPIO Low Power Mode Wakeup Select 2 (GPIOLPMSEL2) Register

The GPIO Low Power Mode Wakeup Select 2 (GPIOLPMSEL2) register is shown and described in the figure and table below.

Figure 4-78. GPIO Low Power Mode Wakeup Select 2 (GPIOLPMSEL2) Register

31	30	29	28	27	26	25	24
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 4-97. GPIO Low Power Mode Wakeup Select 2 (GPIOLPMSEL2) Register Field Descriptions

Bits	Field	Value	Description [(1)]
31-0	GPIO63 - GPIO32		Low Power Mode Wakeup Selection 1. Each bit in this register corresponds to one GPIO port B pin (GPIO32 - GPIO63) .
		0	If the bit is cleared, the signal on the corresponding pin will have no effect on the HALT and STANDBY low power modes.
		1	If the respective bit is set to 1, the signal on the corresponding pin is able to wake the device from both HALT and STANDBY low power modes.

(1) This register is EALLOW protected.

Copyright © 2012–2019, Texas Instruments Incorporated

Internal Memory

This chapter provides information on the RAM and flash memory modules.

Topic **Page**

5.1 RAM Control Module 426
5.2 RAM Control Module Registers 434
5.3 Flash Controller Memory Module 493
5.4 Flash Registers 508

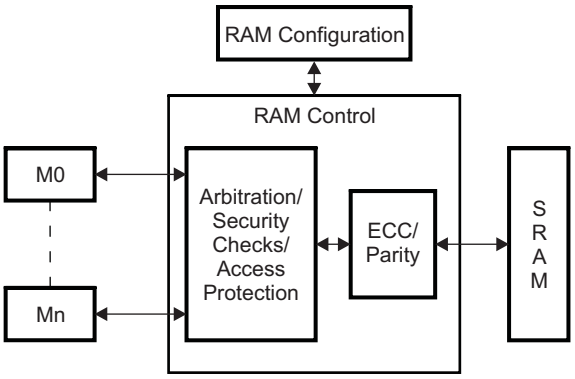
Copyright © 2012–2019, Texas Instruments Incorporated

5.1 RAM Control Module

For these devices, the RAMs have different characteristics. Some are dedicated to a particular master CPU; some are shared between a CPU and its DMA; and, some are shared between both CPUs and both DMAs.

All these RAMs are highly configurable to achieve control for write access and fetch access from different masters. There are also RAMs - called IPC MSGRAMs - that are used for interprocessor communication. All dedicated RAMs are enabled with the ECC feature (both data and addr) and shared RAMs are enabled with the PARITY (both data and addr) feature. Some of the dedicated memories are secure memory as well. Refer to the Security section of the *System Control and Interrupts* chapter for more details. Each RAM has its own controller which takes care of the access protection/security related checks and ECC/Parity features for that RAM. Figure 5-1 shows the configuration of these RAMs.

Figure 5-1. RAM Control



NOTE: All RAMs on Concerto devices are SRAMs.

5.1.1 Functional Description

This section further defines and discusses the dedicated RAMs, shared RAMs, and MSG RAMs on this device.

5.1.1.1 Dedicated RAM

The M3 subsystem has two dedicated RAM blocks: C0 and C1. Only the Cortex-M3 CPU has access to these memories. No other masters (including µDMA) have any access to these memories.

The C28x subsystem has four dedicated RAM blocks: M0, M1, L0, and L1. M0/M1 memories are small blocks of memory which are tightly coupled with the CPU. Only the C28x CPU has access to these memories. No other masters (including DMA) have any access to these memories.

All dedicated RAMs have the ECC feature. All dedicated memories (except for M0/M1) are secure memory and also have the access protection (cpu write protection/cpu fetch protection) feature. Each type of access protection for each RAM block can be enabled/disabled by configuring the specific bit in the RAM block configuration register, allocated to each subsystem (CxDRCR for the M3 subsystem and LxDRCR for the C28x subsystem).

5.1.1.2 Shared RAM

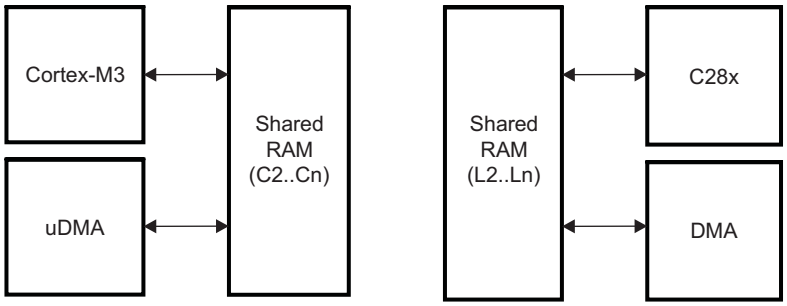
RAM blocks which can be accessed by more than one master (CPU or DMA) are called shared RAMs. On these devices, all such memories are non-secure memory and have the parity feature.

There are different types of shared RAMs. The first type of Shared RAMs are dedicated to each subsystem and have access from the respective CPU and DMA only (see Figure 5-2). Such RAM blocks on the M3 subsystem are mapped to the M3 CPU and M3 µDMA. Similarly for the C28x subsystem, these RAM blocks are mapped to the C28 CPU and C28 DMA.

Copyright © 2012–2019, Texas Instruments Incorporated

All these RAMs have the access protection (cpu write/cpu fetch/dma write) feature. Each type of access protection for each RAM block can be enabled/disabled by configuring the specific bit in the shared RAM configuration registers, allocated to each subsystem (CxSRCR for the M3 subsystem and LxSRCR for the C28x subsystem).

Figure 5-2. Shared RAM (Dedicated to Subsystem)



Another type of shared RAMs (Sx RAMs) is accessible from all the masters and used for data and code storage (see Figure 5-3). These RAMs can be accessed by both CPUs (C28x/M3) and their respective DMA engines (C28 DMA/M3 uDMA). Each shared RAM can be owned by the M3 subsystem or the C28x subsystem based on the configuration of respective bits (one bit for each Sx memory) in the MSxMSEL register. When an Sx RAM block is owned by the M3 subsyteam, the M3 CPU and µDMA have full access to that RAM block, whereas the C28x CPU and DMA have only read access to that RAM block (no fetch/write access).

This register is only mapped to the M3 CPU, which means the master ownership setting for Sx RAMs can be changed by software running on the M3 subsystem. This register can be LOCKED by the user to prevent further updates to the MSxSEL register. Once it's locked, it can be unlocked only by SharedResorceReset. The C28x subsystem has a status register, CSxMSEL, which reflects the ownership status of each Sx RAM block (see Table 5-1).

Figure 5-3. Shared RAM (Shared between Subsystems)

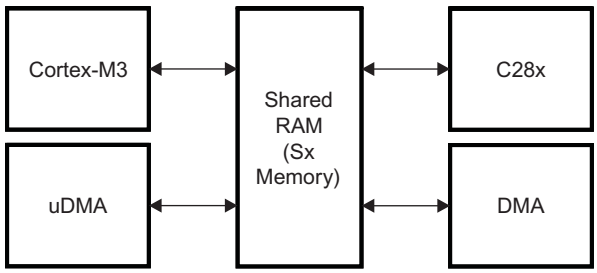


Table 5-1. Master access for Sx RAM (assuming all other protections are disabled)

SxMSEL	M3 Fetch	M3 Read	M3 Write	uDMA Read	uDMA Write	C28x Fetch	C28x Read	C28x Write	DMA Read	DMA Write
0	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No
1	No	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes

Like other shared RAM, these RAMs also have a different level of access protection which can be enabled/disabled by configuring specific bits in the MSxSRCR register (when the RAM block is owned by the M3 subsystem) or the CSxSRCR register (when the RAM block is owned by the C28x subsystem).

Copyright © 2012–2019, Texas Instruments Incorporated

5.1.1.3 MSG RAM

The third type of shared RAMs are called “message RAM” memories (MSG RAMs). There are only two such memory blocks in these devices, specifically used for inter-processor communication (IPC) between the two CPUs. Hence, these are also called as IPC RAMs. The message RAMs will have only CPU and DMA read/write access from one side (M3 or C28), and CPU/DMA read from the other side. For MTOCMSGRAM, M3 CPU and uDMA will have read and write accesses, whereas C28 CPU and DMA will have only read access. Similarly for CTOMMSGRAM, C28 CPU and DMA will have read and write accesses, whereas M3 CPU and µDMA will have only read access. MSG RAMs have protection for DMA/µDMA writes which can be enabled/disabled by configuring specific bits in MSGRAMCR registers, allocated to each subsystem.

5.1.1.4 Access Arbitration

For a shared RAM, multiple accesses can come at a given time. The maximum number of accesses to any Sx RAM at any given time depends on the type of shared RAM. To service one out of multiple accesses at given time, an arbitration scheme needs to be followed. On Concerto devices, a ROUND ROBIN scheme is followed for arbitration. In this scheme all the accesses are serviced in a specific order. This ensures that access from any master is not pending beyond a fix number of cycles in any scenario.

Accesses are serviced on these devices in the following order:

1. M3 CPU Access (RD/WR)
2. M3 uDMA Access (RD/WR)
3. C28 CPU Data Read
4. C28 CPU Write (Data/Program)
5. C28 CPU Program Fetch/Read
6. C28 DMA Access (RD/WR)

Following are some guidelines to follow while allocating code into any of the Sx RAM blocks:

- For least latency on the relocatable NVIC vector table accesses for exception accesses, users should allocate the vector tables in a dedicated RAM block.
- In case of bit-band writes from the M3 CPU, it is possible that the latency for C28 accesses to shared memories will increase, depending on the M3 CPU bit banded accesses. Users must ensure that they partition application memory such that the effect of this does not affect C28 real-time performance requirements.

5.1.1.5 Access Timing

In general on these devices, all RAM blocks have one cycle access time; that is, if there is only one access and no other access is pending to that particular RAM block. However, since there are different types of shared RAMs, where multiple masters can access the same RAM block, that will not always be the case.

Following are access cycle times in different scenarios:

- In case of M3/µDMA, a write access, immediately followed by a read access to the same RAM block, incurs a stall of a single clock cycle.
- M3/µDMA write access can have one extra wait state in case of arbitration and M3/µDMA access does not win during arbitration cycle.
- Max cycle latency for an access to Sx memory from any master is six cycles when M3 is master for that Sx memory. The following are the possible accesses with details of how many cycles each access takes.
  - µDMA access – two cycles (reads or byte writes takes two cycles of C28 clock when M3 clock config is /2 or /4 of C28 clock)
  - M3 byte access – two cycles (reads or byte writes takes two cycles of C28 clock when M3 clock config is /2 or /4 of C28 clock)
  - C28 Read – 1cycle
  - C28 Pread – 1 cycle

Copyright © 2012–2019, Texas Instruments Incorporated

- C28 DMA Read – 1 cycle

- Max cycle latency for an access to Sx memory from any master is seven cycles when C28 is master for that Sx memory. The following are the possible accesses with details of how many cycles each access takes individually.
  - µDMA Read access – two cycles (reads takes two cycles of the C28 clock when M3 clock config is /2 or /4 of C28 clock)
  - M3 Read access – two cycles (reads takes two cycles of the C28 clock when M3 clock config is /2 or /4 of C28 clock)
  - C28 Read – 1 cycle
  - C28 Pread – 1 cycle
  - C28 Write – 1 cycle
  - C28 DMA write/Read – 1 cycle

5.1.1.6 Access Protection

All RAM blocks except for M0/M1 on the C28x subsystem have different levels of protection, based on which masters have access to the RAM block. If a master has access to a particular RAM block, it can always read data from that memory. There is no protection for READ.

The following sections describe the different kind of protection available for RAM blocks on this device.

5.1.1.6.1 CPU Fetch Protection

The M3 or C28x CPU has execution permission from a memory, only if that memory is dedicated to that CPU or its respective subsystem, or if its respective subsystem is master for that memory (in case of Sx memory). When fetch accesses are allowed based on the mastership from a CPU, it can be further protected by setting the FETCHPROTx bit of the specific register to ‘1’. If fetch access is done by CPU to memory where it is protected, a fetch protection violation occurs.

There are two types of fetch protection violations:

- Non-master CPU fecth protection violation (only applicable to Sx memories)
  If a fetch access is made to Sx memory by the non-master CPU, it ’s called a non-master fetch protection violation.
- Master CPU fetch protection violation

If a fetch access is made to a dedicated or shared memory by the master CPU, and FETCHPROTx is set to ‘1’ for that memory, it’s called a master CPU fetch protection violation.

If a fetch protection violation occur on M3, it results into BUSFAULT; whereas, if the violation occurs on C28x it results into ITRAP. A flag is set into the access violation flag register, and the memory address where the violation happened gets latched into the cpu fetch access violation address register. These are dedicated registers for each subsystem.

5.1.1.6.2 CPU Write Protection

The M3 or C28x CPU has write permission to a memory, only if that memory is dedicated to that CPU or its respective subsystem, or its respective subsystem is the master for that memory (in case of Sx memory). When write accesses are allowed based on the mastership from a CPU, it can be further protected by setting the CPUWRPROTx bit of the specific register to ‘1’. If write access is done by CPU to memory where it’s protected, a write protection violation occurs.

There are two types of CPU write protection violations:

- Non-master CPU write protection violation (only applicable to Sx memories)
  If a write access is made to Sx memory by the non-master CPU, it’s called a non-master write protection violation.
- Master CPU write protection violation
  If a write access is made to a dedicated or shared memory by the master CPU and CPUWRPROTx is set to ‘1’ for that memory, it’s called a master CPU write protection violation.

Copyright © 2012–2019, Texas Instruments Incorporated

If a write protection violation occur on M3, write is ignored and BUSFAULT gets generated. If the violation occurs on C28x, write is ignored and access violation interrupt is generated, if enabled in the interrupt enable register. A flag gets set into the cpu access violation flag register, and the memory address where the violation happened gets latched into the cpu write access violation address register. These are dedicated registers for each subsystem.

5.1.1.6.3 DMA Write Protection

The M3 or C28x DMA has write permission to a memory, only if that memory is dedicated to its respective subsystem, or its respective subsystem is the master for that memory (in case of Sx memory). When write accesses from a DMA are allowed based on the mastership,, it can be further protected by setting the DMAWRPROTx bit of specific register to ‘1’. If write access is done by DMA to protected memory, write protection violation occurs.

There are two types of DMA write protection violations:

- Non-master DMA write protection violation (only applicable to Sx memories)
  If a write access is made to Sx memory by the non-master DMA, it’s called a non-master write protection violation.
  If a write access is made to a dedicated or shared memory by the master DMA and DMAWRPROTx is set to ‘1’ for that memory, it’s called a master DMA write protection violation.
- Master DMA write protection violation
  If a write access is made to a dedicated or shared memory by master DMA and DMAWRPROTx is set to ‘1’ for that memory, it’s called a master DMA write protection violation.

If a write protection violation occurs on M3, write is ignored and the dmaerr interrupt gets generated. If a write violation occurs on C28x, write is ignored and an access violation interrupt is generated, if enabled in the interrupt enable register. A flag gets set into the dma access violation flag register, and the memory address where the violation happened gets latched into the dma fetch access violation address register. These are dedicated registers for each subsystem.

All access protections are ignored during debug accesses. Write access to protected memory will go through when it’s done via the debugger, irrespective of the setting of Sx MSEL bits in the MSxMSEL register and CPUWRPROTx setting. There is no protection for the M0 and M1 memories on the C28 subsystem.

5.1.1.7 Memory Error Detection, Correction and Error Handling

These devices have memory error detection and correction features to satisfy safety standards requirements. These requirements warrant the addition of detection mechanisms for finite dangerous failures.

All dedicateds RAMs will support error correction code (ECC) protection and the shared RAMs have parity protection. The ECC scheme used is Single Error Correction Double Error Detection (SECDED). The parity scheme used is even. ECC/Parity will cover the data bits stored in memory as well as address during read and read-modify-write.

ECC/Parity calculation is done inside the RAM control module and then calculated ECC/parity is written into the memory along with the data. ECC/Parity is computed for 16-bit data; hence, for each 32-bit data, there will be three 7-bit ECC codes (or 3-bit parity), two of which are for data and a third one for the address.

Though ECC/parity is calculated for 16-bit data, the controller supports ECC/parity for byte access from masters on the M3 subsystem. In this case, the controller internally reads the complete 16-bit data from the memory, checks the data correctness of the word present in memory, and performs a read–modify–write operation, to store the data as a 16-bit word with ECC/parity code.

5.1.1.7.1 Error Detection and Correction

Error detection is done while reading the data from memory. The error detection is performed for data as well as address. For parity memory, only a single-bit error gets detected whereas, in case of ECC memory, along with a single-bit error, a double-bit error also gets detected. These errors are called correctable error and uncorrectable errors.

Copyright © 2012–2019, Texas Instruments Incorporated

- Parity errors are always uncorrectable errors
- Single-bit ECC errors are correctable errors
- double bit ECC errors are uncorrctbale errors
- Address ECC errors are also uncorrectable errors.

Correctable errors get corrected by RAM Control module and then correct data is given back as read data to master and also written back into the memory. Since RAM Control module performs a read-modify-write operation during byte access from master on M3 subsystem side, there could be error duing read phase. In case this is correctable error, the controller corrects the data and writes into memory but incase of uncorrectable error, appropriate error get generated.

5.1.1.7.2 Error Handling

For each correctable error, the count in the correctable error count register will increment by one. When the value in this count register becomes equal to the value configured into the correctable error threshold register, an interrupt is generated to the respective CPU; that is, if the interrupt is enabled in the correctable interrupt enable register. The user needs to configure the correctable error threshold register based on the system requirement. Also the address for which error occurred, gets latched into the master specific status register and a flag is set. Each of these registers are dedicated for each subsystems.

If there are uncorrectable errors, a BUS-FAULT gets generated for the M3 CPU and a DMA ERROR interrupt gets generated for the µDMA. On the C28x side, an NMI gets generated for the C28x CPU as well as the C28x DMA. Also, in this case, the address for which error occurred, gets latched into the master-specific address status register, and a flag gets set.

Table 5-2 summarizes different error situations that can arise. These need to be handled appropriately in software using the status and interrupt indications provided.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-2. Error Handling in Different Scenarios

Access Type	Error Found In	Error Type	Status Indication	Error Notification
M3/uDMA Byte Write (R-M-W)	Data read from memory	Uncorrectable Error (Parity Error OR Double bit ECC Error)	M3/uDMA Write Address Error Register – Indicates address for which error happened	NVIC Interrupt for uDMA access / bus fault for M3 access
M3/uDMA Byte Write (R-M-W)	Data read from memory	Correctable Error	No	None
Reads	Data read from memory	Uncorrectable Error (Parity Error OR Double bit ECC Error)	Yes - M3 / uDMA / C28 CPU / C28 DMA Read Address Error Register Data returned to M3 CPU / uDMA or C28 CPU / C28 DMA is incorrect	NMI for C28 CPU access NMI for C28 DMA access BUS FAULT for M3 CPU access Interrupt for uDMA access
Reads	Data read from memory	Single data error	Yes - M3 / uDMA / C28 CPU / C28 DMA Read Address Error Register Increment single error counter	Interrupt on error counter reaching the user programmable threshold for single errors
Reads	Address	Address error	Yes - M3 / uDMA / C28 CPU / C28 DMA Read Address Error Register Data returned to M3 CPU / uDMA or C28 CPU / C28 DMA is incorrect	NMI for C28 CPU access NMI for C28 DMA access BUS FAULT for M3 CPU access Interrupt for uDMA access

Notes: In case of uncorrectable error duing fetch on C28x CPU, there is possibility of getting ITRAP before NMI, since garbage instructions enters into C28x pipeline before NMI gets generated.
During debug accesses (RD/WR) correctable as well as uncorrectable errors are masked.

5.1.1.8 Application Test Hooks for Error Detection and Correction

Since error detection and correction logic is part of safety critical logic, safety applications may need to ensure that the logic is working fine always (during run time also). To enable this, a RAMTEST mode is provided. When “RAMTEST” mode is set (ECCPARTEST=1), the ECC/parity logic gets bypassed. For example, while writing the data, only data bit’s will be written into memoy not the ECC/parity bits for data and address. Using this feature an ECC/parity error could be injected into data. Also, the ECC/parity bits for each address location are memory mapped in separate memory locations, accessible directly by the CPU. The memory mapped ECC or parity bits can also be changed by the software. During read back, the ECC / parity logic will detect an error thereby confirming to the application code that ECC/parity logic works fine. If not then application should consider this as an error condition and handle it accordingly.

Notes: The memory map for ECC/parity bits is accessible only when the RAMTEST mode is selected (ECCPARTEST=1) or else reads to these address locations return ‘0.’
In RAMTEST mode, all access to memories (data as well as ECC/partiy) should be done as 32-bit access only.

The following table shows the bit mapping for the ECC/parity bits when they are read in RAMTEST mode using their respective addresses.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-3. Mapping of ECC bits in Read Data from ECC/Parity Address Map

Data Bits Location in Read Data	Content (ECC Memory)
6:0	ECC Code for lower 16 bits of data
7	Not Used
14:8	ECC Code for upper 16 bits of data
15	Not Used
22:16	ECC Code for address
31:23	Not Used

Table 5-4. Mapping of Parity bits in Read Data from ECC/Parity Address Map

Data Bits Location in Read Data	Content (Parity Memory)
0	Parity for lower 16 bits of data
7:1	Not Used
8	Parity for upper 16 bits of data
15:9	Not Used
16	Parity for address
31:17	Not Used

5.1.1.9 RAM Initialization

To ensure that read/fetch (byte write in case of M3/uDMA) from uninitialized RAM locations do not cause ECC or parity errors, the RAM_INIT feature is provided for each memory block. Using this feature, any RAM block can be initialized with 0x0 data and respective ECC/parity bits accordingly. This can be initiated by setting the RAMINIT bit to ‘1’ for the specific RAM block in RTESTINIT registers. To check the status of RAM_INIT, SW should poll for the RAMINITDONE bit for that RAM block in the RINITDONE register to be set. Unless this bit gets set, no access should be made to that RAM memory block.

In case of Sx memory, the CPU of the subsystem, which is configured as the master for the particular Sx RAM block, can only initiate the RAM initialization.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2 RAM Control Module Registers

Table 5-5. M3 RAM Configuration Registers Summary

Register Acronym	Size (x8)	Offset (x8)	Protection	Reset Source	Register Description
CxDRCR1	4	0x0	Protected	M3	Cx DEDRAM Configuration Register 1
CxSRCR1	4	0x8	Protected	M3	Cx SHRAM Configuration Register 1
MSxMSEL	4	0x10	Protected + Lock	Shared	Sx SHRAM Master Select Register
MSxSRCR1	4	0x20	Protected	M3	M3 Sx SHRAM Configuration Register 1
MSxSRCR2	4	0x24	Protected	M3	M3 Sx SHRAM Configuration Register 2
MTOCMSGRCR	4	0x30	Protected	M3	M3TOC28_MSG_RAM Configuration Register
CxRTESTINIT1	4	0x40	Protected	M3	Cx RAM Test and Initialization Register 1
MSxRTESTINIT1	4	0x50	Protected	M3	M3 Sx RAM Test and Initialization Register 1
MTOCRTESTINIT	4	0x60	Protected	M3	MTOC_MSG_RAM Test and Initialization Register
CxRINITDONE1	4	0x70		M3	Cx RAM INITDONE Register 1
MSxRINITDONE1	4	0x78		M3	M3 Sx RAM INITDONE Register 1
MTOCRINITDONE	4	0x88		M3	MTOC_MSG_RAM INITDONE Register 1

Table 5-6. M3 RAM Error Registers Summary

Register Acronym	Size (x8)	Offset (x8)	Protection	Reset Source	Register Description
MCUNCWEADDR	4	0x0		M3	M3 CPU Uncorrectable Write Error Address Register
MDUNCWEADDR	4	0x4		M3	M3 µDMA Uncorrectable Write Error Address Register
MCUNCREADDR	4	0x8		M3	M3 CPU Uncorrectable Read Error Address Register
MDUNCREADDR	4	0xC		M3	M3 µDMA Uncorrectable Read Error Address Register
MCPUCREADDR	4	0x10		M3	M3 CPU Corrected Read Error Address Register
MDMACREADDR	4	0x14		M3	M3 µDMA Corrected Read Error Address Register
MUEFLG	4	0x20		M3	M3 Uncorrectable Error Flag Register
MUEFRC	4	0x24		M3	M3 Uncorrectable Error Force Register
MUECLR	4	0x28		M3	M3 Uncorrectable Error Flag Clear Register
MCECNTR	4	0x2C		M3	M3 Corrected Error Counter Register
MCETRES	4	0x30		M3	M3 Corrected Error Threshold Register
MCEFLG	4	0x38		M3	M3 Corrected Error Threshold Exceeded Flag Register
MCEFRC	4	0x3C		M3	M3 Corrected Error Threshold Exceeded Force Register
MCECLR	4	0x40		M3	M3 Corrected Error Threshold Exceeded Flag Clear Register
MCEIE	4	0x44		M3	M3 Single Error Interrupt Enable Register
MNMAVFLG	4	0x50		M3	Non-Master Access Violation Flag Register
MNMAVCLR	4	0x58		M3	Non-Master Access Violation Flag Clear Register
MMAVFLG	4	0x60		M3	Master Access Violation Flag Register
MMAVCLR	4	0x68		M3	Master Access Violation Flag Clear Register
MNMWRAVADDR	4	0x70		M3	Non-Master CPU Write Access Violation Address Register

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-6. M3 RAM Error Registers Summary (continued)

Register Acronym	Size (x8)	Offset (x8)	Protection	Reset Source	Register Description
MNMDMAWRAVADDR	4	0x74		M3	Non-Master DMA Write Access Violation Address Register
MNMFAVADDR	4	0x78		M3	Non-Master CPU Fetch Access Violation Address Register
MMWRAVADDR	4	0x80		M3	Master CPU Write Access Violation Address Register
MMDMAWRAVADDR	4	0x84		M3	Master DMA Write Access Violation Address Register
MMFAVADDR	4	0x88		M3	Master CPU Fetch Access Violation Address Register

Table 5-7. C28x RAM Configuration Registers Summary

Register Acronym	Size (x8)	Offset (x16)	Protection	Reset Source	Register Description
LxDRCR1	4	0x0	EALLOW	C28x	Lx DEDRAM Configuration Register 1
LxSRCR1	4	0x4	EALLOW	C28x	Lx SHRAM Configuration Register 1
CSxMSEL	4	0x8	EALLOW	Shared	C28x Sx SHRAM Master Select Register
CSxSRCR1	4	0x10	EALLOW	C28x	C28x Sx SHRAM Configuration Register 1
CSxSRCR2	4	0x12	EALLOW	C28x	C28x Sx SHRAM Configuration Register 2
CTOMMSGRCR	4	0x1A	EALLOW	C28x	C28TOC28_MSG_RAM Configuration Register
C28RTESTINIT	4	0x20	EALLOW	C28x	M0, M1 and C28T0C28_MSG_RAM Test and Initialization Register
CLxRTESTINIT1	4	0x22	EALLOW	C28x	Lx RAM Test and Initialization Register 1
CSxRTESTINIT1	4	0x26	EALLOW	C28x	C28x Sx RAM Test and Initialization Register 1
C28RINITDONE	4	0x30		C28x	M0, M1 and C28T0M3_MSG_RAM INIT Done Register
CLxRINITDONE1	4	0x32		C28x	C28x Lx RAM_INIT_DONE Register 1
CSxRINITDONE1	4	0x36		C28x	C28x Sx RAM_INIT_DONE Register 1

Table 5-8. C28x RAM Error Registers Summary

Register Acronym	Size (x8)	Offset (x16)	Protection	Reset Source	Register Description
CCUNCREADDR	4	0x0		C28x	C28x CPU Uncorrectable Read Error Address Register
CDUNCREADDR	4	0x2		C28x	C28x DMA Uncorrectable Read Error Address Register
CCPUCREADDR	4	0x4		C28x	C28x CPU Corrected Read Error Address Register
CDMACREADDR	4	0x6		C28x	C28x DMA Corrected Read Error Address Register
CUEFLG	4	0x8		C28x	C28x Uncorrectable Error Flag Register
CUEFRC	4	0x0A		C28x	C28x Uncorrectable Error Force Register
CUECLR	4	0x0C		C28x	C28x Uncorrectable Error Flag Clear Register
CCECNTR	4	0x10		C28x	C28x Corrected Error Counter Register
CCETRES	4	0x12		C28x	C28x Corrected Error Threshold Register
CCEFLG	4	0x14		C28x	C28x Corrected Error Threshold Exceeded Flag Register
CEFRC	4	0x16		C28x	C28x Corrected Error Threshold Exceeded Force Register

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-8. C28x RAM Error Registers Summary (continued)

Register Acronym	Size (x8)	Offset (x16)	Protection	Reset Source	Register Description
CCECLR	4	0x18		C28x	C28x Corrected Error Threshold Exceeded Flag Clear Register
CCEIE	4	0x1A		C28x	C28x Single Error Interrupt Enable Register
CNMAVFLG	4	0x20		C28x	Non-Master Access Violation Flag Register
CNMAVFRC	4	0x22		C28x	Non-Master Access Violation Force Register
CNMAVCLR	4	0x24		C28x	Non-Master Access Violation Flag Clear Register
CNMAVIE	4	0x26		C28x	Non-Master Access Violation Interrupt Enable Register
CMAVFLG	4	0x28		C28x	Master Access Violation Flag Register
CMAVFRC	4	0x2A		C28x	Master Access Violation Force Register
CMAVCLR	4	0x2C		C28x	Master Access Violation Flag Clear Register
CMAVIE	4	0x2E		C28x	Master Access Violation Interrupt Enable Register
CNMWRAVADDR	4	0x30		C28x	Non-Master CPU Write Access Violation Address Register
CNMDMAWRAVADDR	4	0x32		C28x	Non-Master DMA Write Access Violation Address Register
CNMFAVADDR	4	0x34		C28x	Non-Master CPU Fetch Access Violation Address Register
CMWRAVADDR	4	0x38		C28x	Master CPU Write Access Violation Address Register
CMDMAWRAVADDR	4	0x3A		C28x	Master DMA Write Access Violation Address Register
CMFAVADDR	4	0x3C		C28x	Master CPU Fetch Access Violation Address Register

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1 M3 RAM Configuration Registers

5.2.1.1 Cx DEDRAM Configuration Register 1 (CxDRCR1)

Figure 5-4. Cx DEDRAM Configuration Register 1 (CxDRCR1)

31	16
Reserved	
R-0	

15	11	10	9	8
Reserved		CPUWRPROTC1	Reserved	FETCHPROTC1
R-0		R/W-0	R-0	R/W-0

7	3	2	1	0
Reserved		CPUWRPROTC0	Reserved	FETCHPROTC0
R-0		R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-9. Cx DEDRAM Configuration Register 1 (CxDRCR1) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved		Reserved
10	CPUWRPROTC1		CPU Write Protection C1
		0	M3 CPU write allowed to C1 RAM block.
		1	M3 CPU write not allowed to C1 RAM block.
9	Reserved		Reserved
8	FETCHPROTC1		CPU Fetch Protection C1
		0	M3 CPU Fetch allowed from C1 RAM block.
		1	M3 CPU Fetch not allowed from C1 RAM block.
7-3	Reserved		Reserved
2	CPUWRPROTC0		CPU Write Protection C0
		0	M3 CPU write allowed to C0 RAM block.
		1	M3 CPU write not allowed to C0 RAM block.
1	Reserved		Reserved
0	FETCHPROTC0		CPU Fetch Protection C0
		0	M3 CPU Fetch allowed from C0 RAM block.
		1	M3 CPU Fetch not allowed from C0 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1.2 Cx SHRAM Configuration Register 1 (CxSRCR1)

Figure 5-5. Cx SHRAM Configuration Register 1 (CxSRCR1)

31				16
Reserved				
R-0				
15	11	10	9	8
Reserved		CPUWRPROTC3	DMAWRPROTC3	FETCHPROTC3
R-0		R/W-0	R/W-0	R/W-0
7	3	2	1	0
Reserved		CPUWRPROTC2	DMAWRPROTC2	FETCHPROTC2
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 5-10. Cx SHRAM Configuration Register 1 (CxSRCR1) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved		Reserved
10	CPUWRPROTC3		CPU Write Protection C3
		0	M3 CPU write allowed to C3 RAM block.
		1	M3 CPU write not allowed to C3 RAM block.
9	DMAWRPROTC3		µDMA Write Protection C3
		0	M3 µDMA write allowed to C3 RAM block.
		1	M3 µDMA write not allowed to C3 RAM block.
8	FETCHPROTC3		CPU Fetch Protection C3
		0	M3 CPU Fetch allowed from C3 RAM block.
		1	M3 CPU Fetch not allowed from C3 RAM block.
7-3	Reserved		Reserved
2	CPUWRPROTC2		CPU Write Protection C2
		0	M3 CPU write allowed to C2 RAM block.
		1	M3 CPU write not allowed to C2 RAM block.
1	DMAWRPROTC2		µDMA Write Protection C2
		0	M3 µDMA write allowed to C2 RAM block.
		1	M3 µDMA write not allowed to C2 RAM block.
0	FETCHPROTC2		CPU Fetch Protection C2
		0	M3 CPU Fetch allowed from C2 RAM block.
		1	M3 CPU Fetch not allowed from C2 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1.3 Sx SHRAM Master Select Register (MSxMSEL)

Figure 5-6. Sx SHRAM Master Select Register (MSxMSEL)

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

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
S7MSEL	S6MSEL	S5MSEL	S4MSEL	S3MSEL	S2MSEL	S1MSEL	S0MSEL
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-11. Sx SHRAM Master Select Register (MSxMSEL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7	S7MSEL		Master Ownership for S7 RAM Block
		0	M3 subsystem is master for S7 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S7 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
6	S6MSEL		Master Ownership for S6 RAM Block
		0	M3 subsystem is master for S6 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S6 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
5	S5MSEL		Master Ownership for S5 RAM Block
		0	M3 subsystem is master for S5 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S5 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
4	S4MSEL		Master Ownership for S4 RAM Block
		0	M3 subsystem is master for S4 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S4 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
3	S3MSEL		Master Ownership for S3 RAM Block
		0	M3 subsystem is master for S3 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S3 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
2	S2MSEL		Master Ownership for S2 RAM Block
		0	M3 subsystem is master for S2 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S2 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
1	S1MSEL		Master Ownership for S1 RAM Block
		0	M3 subsystem is master for S1 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S1 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-11. Sx SHRAM Master Select Register (MSxMSEL) Field Descriptions (continued)

Bit	Field	Value	Description
0	S0MSEL		Master Ownership for S0 RAM Block
		0	M3 subsystem is master for S0 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S0 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.

5.2.1.4 M3 Sx SHRAM Configuration Register 1 (MSxSRCR1)

Figure 5-7. M3 Sx SHRAM Configuration Register 1 (MSxSRCR1)

31	27	26	25	24
Reserved		CPUWRPROTS3	DMAWRPROTS3	FETCHPROTS3
R-0		R/W-0	R/W-0	R/W-0
23	19	18	17	16
Reserved		CPUWRPROTS2	DMAWRPROTS2	FETCHPROTS2
R-0		R/W-0	R/W-0	R/W-0
15	11	10	9	8
Reserved		CPUWRPROTS1	DMAWRPROTS1	FETCHPROTS1
R-0		R/W-0	R/W-0	R/W-0
7	3	2	1	0
Reserved		CPUWRPROTS0	DMAWRPROTS0	FETCHPROTS0
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-12. M3 Sx SHRAM Configuration Register 1 (MSxSRCR1) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26	CPUWRPROTS3		CPU Write Protection S3
		0	M3 CPU write allowed to S3 RAM block.
		1	M3 CPU write not allowed to S3 RAM block.
25	DMAWRPROTS3		µDMA Write Protection S3
		0	M3 µDMA write allowed to S3 RAM block.
		1	M3 µDMA write not allowed to S3 RAM block.
24	FETCHPROTS3		CPU Fetch Protection S3
		0	M3 CPU Fetch allowed from S3 RAM block.
		1	M3 CPU Fetch not allowed from S3 RAM block.
23-19	Reserved		Reserved
18	CPUWRPROTS2		CPU Write Protection S2
		0	M3 CPU write allowed to S2 RAM block.
		1	M3 CPU write not allowed to S2 RAM block.
17	DMAWRPROTS2		µDMA Write Protection S2
		0	M3 µDMA write allowed to S2 RAM block.
		1	M3 µDMA write not allowed to S2 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-12. M3 Sx SHRAM Configuration Register 1 (MSxSRCR1) Field Descriptions (continued)

Bit	Field	Value	Description
16	FETCHPROTS2		CPU Fetch Protection S2
		0	M3 CPU Fetch allowed from S2 RAM block.
		1	M3 CPU Fetch not allowed from S2 RAM block.
15-11	Reserved		Reserved
10	CPUWRPROTS1		CPU Write Protection S1
		0	M3 CPU write allowed to S1 RAM block.
		1	M3 CPU write not allowed to S1 RAM block.
9	DMAWRPROTS1		µDMA Write Protection S1
		0	M3 µDMA write allowed to S1 RAM block.
		1	M3 µDMA write not allowed to S1 RAM block.
8	FETCHPROTS1		CPU Fetch Protection S1
		0	M3 CPU Fetch allowed from S1 RAM block.
		1	M3 CPU Fetch not allowed from S1 RAM block.
7-3	Reserved		Reserved
2	CPUWRPROTS0		CPU Write Protection S0
		0	M3 CPU write allowed to S0 RAM block.
		1	M3 CPU write not allowed to S0 RAM block.
1	DMAWRPROTS0		µDMA Write Protection S0
		0	M3 µDMA write allowed to S0 RAM block.
		1	M3 µDMA write not allowed to S0 RAM block.
0	FETCHPROTS0		CPU Fetch Protection S0
		0	M3 CPU Fetch allowed from S0 RAM block.
		1	M3 CPU Fetch not allowed from S0 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1.5 M3 Sx SHRAM Configuration Register 2 (MSxSRCR2)

Figure 5-8. M3 Sx SHRAM Configuration Register 2 (MSxSRCR2)

31	27	26	25	24
Reserved		CPUWRPROTS7	DMAWRPROTS7	FETCHPROTS7
R-0		R/W-0	R/W-0	R/W-0
23	19	18	17	16
Reserved		CPUWRPROTS6	DMAWRPROTS6	FETCHPROTS6
R-0		R/W-0	R/W-0	R/W-0
15	11	10	9	8
Reserved		CPUWRPROTS5	DMAWRPROTS5	FETCHPROTS5
R-0		R/W-0	R/W-0	R/W-0
7	3	2	1	0
Reserved		CPUWRPROTS4	DMAWRPROTS4	FETCHPROTS4
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 5-13. M3 Sx SHRAM Configuration Register 2 (MSxSRCR2) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26	CPUWRPROTS7		CPU Write Protection S7
		0	M3 CPU write allowed to S7 RAM block.
		1	M3 CPU write not allowed to S7 RAM block.
25	DMAWRPROTS7		µDMA Write Protection S7
		0	M3 µDMA write allowed to S7 RAM block.
		1	M3 µDMA write not allowed to S7 RAM block.
24	FETCHPROTS7		CPU Fetch Protection S7
		0	M3 CPU Fetch allowed from S7 RAM block.
		1	M3 CPU Fetch not allowed from S7 RAM block.
23-19	Reserved		Reserved
18	CPUWRPROTS6		CPU Write Protection S6
		0	M3 CPU write allowed to S6 RAM block.
		1	M3 CPU write not allowed to S6 RAM block.
17	DMAWRPROTS6		µDMA Write Protection S6
		0	M3 µDMA write allowed to S6 RAM block.
		1	M3 µDMA write not allowed to S6 RAM block.
16	FETCHPROTS6		CPU Fetch Protection S6
		0	M3 CPU Fetch allowed from S6 RAM block.
		1	M3 CPU Fetch not allowed from S6 RAM block.
15-11	Reserved		Reserved
10	CPUWRPROTS5		CPU Write Protection S5
		0	M3 CPU write allowed to S5 RAM block.
		1	M3 CPU write not allowed to S5 RAM block.
9	DMAWRPROTS5		µDMA Write Protection S5
		0	M3 µDMA write allowed to S5 RAM block.
		1	M3 µDMA write not allowed to S5 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-13. M3 Sx SHRAM Configuration Register 2 (MSxSRCR2) Field Descriptions (continued)

Bit	Field	Value	Description
8	FETCHPROTS5		CPU Fetch Protection S5
		0	M3 CPU Fetch allowed from S5 RAM block.
		1	M3 CPU Fetch not allowed from S5 RAM block.
7-3	Reserved		Reserved
2	CPUWRPROTS4		CPU Write Protection S4
		0	M3 CPU write allowed to S4 RAM block.
		1	M3 CPU write not allowed to S4 RAM block.
1	DMAWRPROTS4		µDMA Write Protection S4
		0	M3 µDMA write allowed to S4 RAM block.
		1	M3 µDMA write not allowed to S4 RAM block.
0	FETCHPROTS4		CPU Fetch Protection S4
		0	M3 CPU Fetch allowed from S4 RAM block.
		1	M3 CPU Fetch not allowed from S4 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1.6 M3TOC28_MSG_RAM Configuration Register (MTOCMSGRCR)

Figure 5-9. M3TOC28_MSG_RAM Configuration Register (MTOCMSGRCR)

31	2	1	0
Reserved		DMAWRPROT	Rsvd
R-0		R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-14. M3TOC28_MSG_RAM Configuration Register (MTOCMSGRCR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	DMAWRPROT		µDMA Write Protection
		0	M3 µDMA write allowed to MTOC_MSG_RAM.
		1	M3 µDMA write not allowed to MTOC_MSG_RAM.
0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1.7 Cx RAM Test and Initialization Register 1 (CxRTESTINIT1)

Figure 5-10. Cx RAM Test and Initialization Register 1 (CxRTESTINIT1)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
ECCPARTESTC3	RAMINITC3	ECCPARTESTC2	RAMINITC2	ECCPARTESTC1	RAMINITC1	ECCPARTESTC0	RAMINITC0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-15. Cx RAM Test and Initialization Register 1 (CxRTESTINIT1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7	ECCPARTESTC3		Enable/Disable RAMTEST Feature for C3 RAM Block.
		0	RAMTEST feature is disabled for C3 RAM block.
		1	RAMTEST feature is enabled for C3 RAM block. ECC/parity logic is bypassed for memory accesses.
6	RAMINITC3		RAM Initialization C3. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of C3 RAM block with data 0x0 and corresponding data an address ECC/parity bits.
5	ECCPARTESTC2		Enable/Disable RAMTEST Feature for C2 RAM Block
		0	RAMTEST feature is disabled for C2 RAM block.
		1	RAMTEST feature is enabled for C2 RAM block. ECC/parity logic is bypassed for memory accesses.
4	RAMINITC2		RAM Initialization C2. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of C2 RAM block with data 0x0 and corresponding data an address ECC/parity bits.
3	ECCPARTESTC1		Enable/Disable RAMTEST Feature for C1 RAM Block
		0	RAMTEST feature is disabled for C1 RAM block.
		1	RAMTEST feature is enabled for C1 RAM block. ECC/parity logic is bypassed for memory accesses.
2	RAMINITC1		RAM Initialization C1. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of C1 RAM block with data 0x0 and corresponding data an address ECC/parity bits.
1	ECCPARTESTC0		Enable/Disable RAMTEST Feature for C0 RAM Block
		0	RAMTEST feature is disabled for C0 RAM block.
		1	RAMTEST feature is enabled for C0 RAM block. ECC/parity logic is bypassed for memory accesses.
0	RAMINITC0		RAM Initialization C0. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of C0 RAM block with data 0x0 and corresponding data an address ECC/parity bits.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1.8 M3 Sx RAM Test and Initialization Register 1 (MSxRTESTINIT1)

Figure 5-11. M3 Sx RAM Test and Initialization Register 1 (MSxRTESTINIT1)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
ECCPARTESTS7	RAMINITS7	ECCPARTESTS6	RAMINITS6	ECCPARTESTS5	RAMINITS5	ECCPARTESTS4	RAMINITS4
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
ECCPARTESTS3	RAMINITS3	ECCPARTESTS2	RAMINITS2	ECCPARTESTS1	RAMINITS1	ECCPARTESTS0	RAMINITS0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-16. M3 Sx RAM Test and Initialization Register 1 (MSxRTESTINIT1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	ECCPARTESTS7		Enable/Disable RAMTEST Feature for S7 RAM Block if M3 Subsystem is Master for S7 RAM Block
		0	RAMTEST feature is disabled for S7 RAM block.
		1	RAMTEST feature is enabled for S7 RAM block. ECC/parity logic is bypassed for memory accesses.
14	RAMINITS7		RAM Initialization S7. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S7 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicabale only if M3 subsystem is master for S7 memory.
13	ECCPARTESTS6		Enable/Disable RAMTEST Feature for S6 RAM Block if M3 Subsystem is Master for S6 RAM Block
		0	RAMTEST feature is disabled for S6 RAM block.
		1	RAMTEST feature is enabled for S6 RAM block. ECC/parity logic is bypassed for memory accesses.
12	RAMINITS6		RAM Initialization S6. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S6 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicabale only if M3 subsystem is master for S6 memory.
11	ECCPARTESTS5		Enable/Disable RAMTEST Feature for S5 RAM Block if M3 Subsystem is Master for S5 RAM Block
		0	RAMTEST feature is disabled for S5 RAM block.
		1	RAMTEST feature is enabled for S5 RAM block. ECC/parity logic is bypassed for memory accesses.
10	RAMINITS5		RAM Initialization S5. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S5 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicabale only if M3 subsystem is master for S5 memory.
9	ECCPARTESTS4		Enable/Disable RAMTEST Feature for S4 RAM Block if M3 Subsystem is Master for S4 RAM Block
		0	RAMTEST feature is disabled for S4 RAM block.
		1	RAMTEST feature is enabled for S4 RAM block. ECC/parity logic is bypassed for memory accesses.
8	RAMINITS4		RAM Initialization S4. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S4 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicabale only if M3 subsystem is master for S4 memory.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-16. M3 Sx RAM Test and Initialization Register 1 (MSxRTESTINIT1) Field Descriptions (continued)

Bit	Field	Value	Description
7	ECCPARTESTS3		Enable/Disable RAMTEST Feature for S3 RAM Block if M3 Subsystem is Master for S3 RAM Block
		0	RAMTEST feature is disabled for S3 RAM block.
		1	RAMTEST feature is enabled for S3 RAM block. ECC/parity logic is bypassed for memory accesses.
6	RAMINITS3		RAM Initialization S3. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S3 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicabale only if M3 subsystem is master for S3 memory.
5	ECCPARTESTS2		Enable/Disable RAMTEST Feature for S2 RAM Block if M3 Subsystem is Master for S2 RAM Block
		0	RAMTEST feature is disabled for S2 RAM block.
		1	RAMTEST feature is enabled for S2 RAM block. ECC/parity logic is bypassed for memory accesses.
4	RAMINITS2		RAM Initialization S2. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S2 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicabale only if M3 subsystem is master for S2 memory.
3	ECCPARTESTS1		Enable/Disable RAMTEST Feature for S1 RAM Block if M3 Subsystem is Master for S1 RAM Block
		0	RAMTEST feature is disabled for S1 RAM block.
		1	RAMTEST feature is enabled for S1 RAM block. ECC/parity logic is bypassed for memory accesses.
2	RAMINITS1		RAM Initialization S1. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S1 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicabale only if M3 subsystem is master for S1 memory.
1	ECCPARTESTS0		Enable/Disable RAMTEST Feature for S0 RAM Block if M3 Subsystem is Master for S0 RAM Block
		0	RAMTEST feature is disabled for S0 RAM block.
		1	RAMTEST feature is enabled for S0 RAM block. ECC/parity logic is bypassed for memory accesses.
0	RAMINITS0		RAM Initialization S0.
		0	No action taken.
		1	Initialize all address locations of S0 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicabale only if M3 subsystem is master for S0 memory.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1.9 MTOC_MSG_RAM Test and Initialization Register (MTOCRTESTINIT)

Figure 5-12. MTOC_MSG_RAM Test and Initialization Register (MTOCRTESTINIT)

31	2	1	0
Reserved		ECCPARTEST MTOCMSGRAM	RAMINIT MTOCMSGRAM
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-17. MTOC_MSG_RAM Test and Initialization Register (MTOCRTESTINIT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	ECCPARTEST MTOCMSGRAM		Enable/Disable RAMTEST Feature for MTOC_MSG_RAM Block
		0	RAMTEST feature is disabled for MTOC_MSG_RAM block.
		1	RAMTEST feature is enabledd for MTOC_MSG_RAM block.
0	RAMINIT MTOCMSGRAM		RAM Initialization for MTOC_MSG_RAM Block. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of MTOC_MSG_RAM block with data 0x0 and corresponding data and address ECC/parity bits.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1.10 Cx RAM INITDONE Register 1 (CxRINITDONE1)

Figure 5-13. Cx RAM INITDONE Register 1 (CxRINITDONE1)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved	RAMINITDONEC3	Reserved	RAMINITDONEC2	Reserved	RAMINITDONEC1	Reserved	RAMINITDONEC0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-18. Cx RAM INITDONE Register 1 (CxRINITDONE1) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved		Reserved
6	RAMINITDONEC3		RAM Initialization Process Status when RAMINIT is Set for C3 RAM Block
		0	RAM initialization is not finished for C3 RAM block.
		1	RAM initialization is done for C3 RAM block. C3 RAM can be accessed by M3 CPU.
			This status bit gets cleared when the RAMINIT bit is set for C3 RAM block.
5	Reserved		Reserved
4	RAMINITDONEC2		RAM Initialization Process Status when RAMINIT is Set for C2 RAM Block
			RAM initialization is not finished for C2 RAM block.
		0	RAM initialization is done for C2 RAM block. C2 RAM can be accessed by M3 CPU.
		1	This status bit gets cleared when the RAMINIT bit is set for C2 RAM block.
3	Reserved		Reserved
2	RAMINITDONEC1		RAM Initialization Process Status when RAMINIT is Set for C1 RAM Block
			RAM initialization is not finished for C1 RAM block.
		0	RAM initialization is done for C1 RAM block. C1 RAM can be accessed by M3 CPU.
		1	This status bit gets cleared when the RAMINIT bit is set for C1 RAM block.
1	Reserved		Reserved
0	RAMINITDONEC0		RAM Initialization Process Status when RAMINIT is Set for C0 RAM Block
			RAM initialization is not finished for C0 RAM block.
		0	RAM initialization is done for C0 RAM block. C0 RAM can be accessed by M3 CPU.
		1	This status bit gets cleared when the RAMINIT bit is set for C0 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1.11 M3 Sx RAM INITDONE Register 1 (MSxRINITDONE1)

Figure 5-14. M3 Sx RAM INITDONE Register 1 (MSxRINITDONE1)

31	30	29	28	27	26	25	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved	RAMINITDONES7	Reserved	RAMINITDONES6	Reserved	RAMINITDONES5	Reserved	RAMINITDONES4
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
Reserved	RAMINITDONES3	Reserved	RAMINITDONES2	Reserved	RAMINITDONES1	Reserved	RAMINITDONES0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-19. M3 Sx RAM INITDONE Register 1 (MSxRINITDONE1) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14	RAMINITDONES7		RAM Initialization Process Status when RAMINIT is Set for S7 RAM Block
		0	RAM initialization is not finished for S7 RAM block.
		1	RAM initialization is done for S7 RAM block. S7 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S7 RAM block.
13	Reserved		Reserved
12	RAMINITDONES6		RAM Initialization Process Status when RAMINIT is Set for S6 RAM Block
		0	RAM initialization is not finished for S6 RAM block.
		1	RAM initialization is done for S6 RAM block. S6 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S6 RAM block.
11	Reserved		Reserved
10	RAMINITDONES5		RAM Initialization Process Status when RAMINIT is Set for S5 RAM Block
		0	RAM initialization is not finished for S5 RAM block.
		1	RAM initialization is done for S7\5 RAM block. S5 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S5 RAM block.
9	Reserved		Reserved
8	RAMINITDONES4		RAM Initialization Process Status when RAMINIT is Set for S4 RAM Block
		0	RAM initialization is not finished for S4 RAM block.
		1	RAM initialization is done for S4 RAM block. S4 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S4 RAM block.
7	Reserved		Reserved
6	RAMINITDONES3		RAM Initialization Process Status when RAMINIT is Set for S3 RAM Block
		0	RAM initialization is not finished for S3 RAM block.
		1	RAM initialization is done for S3 RAM block. S3 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S3 RAM block.
5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-19. M3 Sx RAM INITDONE Register 1 (MSxRINITDONE1) Field Descriptions (continued)

Bit	Field	Value	Description
4	RAMINITDONES2		RAM Initialization Process Status when RAMINIT is Set for S2 RAM Block
		0	RAM initialization is not finished for S2 RAM block.
		1	RAM initialization is done for S2 RAM block. S2 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S2 RAM block.
3	Reserved		Reserved
2	RAMINITDONES1		RAM Initialization Process Status when RAMINIT is Set for S1 RAM Block
		0	RAM initialization is not finished for S1 RAM block.
		1	RAM initialization is done for S1 RAM block. S1 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S1 RAM block.
1	Reserved		Reserved
0	RAMINITDONES0		RAM Initialization Process Status when RAMINIT is Set for S0 RAM Block
		0	RAM initialization is not finished for S0 RAM block.
		1	RAM initialization is done for S0 RAM block. S0 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S0 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.1.12 MTOC_MSG_RAM INITDONE Register (MTOCRINITDONE)

Figure 5-15. MTOC_MSG_RAM INITDONE Register (MTOCRINITDONE)

31	1	0
Reserved		RAMINITDONE MTOCMSGRAM
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-20. MTOC_MSG_RAM INITDONE Register (MTOCRINITDONE) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	RAMINITDONE MTOCMSGRAM		RAM Initialization Status when RAMINIT is Set for MTOC_MSG_RAM Block
		0	RAM initialization is not finished for MTOC_MSG_RAM block.
		1	RAM initialization is done for MTOC_MSG_RAM block. MTOC_MSG_RAM can be accessed by M3 CPU/µDMA.
			This status bit gets cleared when the RAMINIT bit is set for CMTOC_MSG_RAM block.

5.2.2 M3 RAM Error Registers

5.2.2.1 M3 CPU Uncorrectable Write Error Address Register (MCUNCWEADDR)

Figure 5-16. M3 CPU Uncorrectable Write Error Address Register (MCUNCWEADDR)

31	0
MCUNCWEADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-21. M3 CPU Uncorrectable Write Error Address Register (MCUNCWEADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MCUNCWEADDR		This register contains the address where uncorrectable error occurs during M3 CPU byte writes. Only the address coresponding to the last error is stored.

5.2.2.2 M3 µDMA Uncorrectable Write Error Address Register (MDUNCWEADDR)

Figure 5-17. M3 µDMA Uncorrectable Write Error Address Register (MDUNCWEADDR)

31	0
MDUNCWEADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-22. M3 µDMA Uncorrectable Write Error Address Register (MDUNCWEADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MDUNCWEADDR		This register contains the address where uncorrectable error occurs during M3 µDMA byte writes. Only the address coresponding to the last error is stored.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.3 M3 CPU Uncorrectable Read Error Address Register (MCUNCREADDR)

Figure 5-18. M3 CPU Uncorrectable Read Error Address Register (MCUNCREADDR)

31	0
MCUNCREADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-23. M3 CPU Uncorrectable Read Error Address Register (MCUNCREADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MCUNCREADDR		This register contains the address where uncorrectable error occurs during M3 CPU data read or fetch. Only the address coresponding to the last error is stored.

5.2.2.4 M3 µDMA Uncorrectable Read Error Address Register (MDUNCREADDR)

Figure 5-19. M3 µDMA Uncorrectable Read Error Address Register (MDUNCREADDR)

31	0
MDUNCREADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-24. M3 µDMA Uncorrectable Read Error Address Register (MDUNCREADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MDUNCREADDR		This register contains the address where uncorrectable error occurs during M3 µDMA data read. Only the address coresponding to the last error is stored.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.5 M3 CPU Corrected Read Error Address Register (MCPUCREADDR)

Figure 5-20. M3 CPU Corrected Read Error Address Register (MCPUCREADDR)

31	0
MCPUCREADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-25. M3 CPU Corrected Read Error Address Register (MCPUCREADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MCPUCREADDR		This register contains the address where correctable error occurs during M3 CPU data read or fetch. Only the address coresponding to the last error is stored.

5.2.2.6 M3 µDMA Corrected Read Error Address Register (MDMACREADDR)

Figure 5-21. M3 µDMA Corrected Read Error Address Register (MDMACREADDR)

31	0
MDMACREADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-26. M3 µDMA Corrected Read Error Address Register (MDMACREADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MDMACREADDR		This register contains the address where correctable error occurs during M3 µDMA data read. Only the address coresponding to the last error is stored.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.7 M3 Uncorrectable Error Flag Register (MUEFLG)

Figure 5-22. M3 Uncorrectable Error Flag Register (MUEFLG)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		UDMARE	M3CPURE	UDMAWE	M3CPUWE
R-0		R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-27. M3 Uncorrectable Error Flag Register (MUEFLG) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reserved
3	UDMARE		M3 µDMA Uncorrectable Read Error Status Flag .
		0	No M3 µDMA uncorrectable read error occurred.
		1	M3 µDMA uncorrectable read error occurred.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MUECLR register.
2	M3CPURE		M3 CPU Uncorrectable Read Error Status Flag.
		0	No M3 CPU uncorrectable read error occurred.
		1	M3 CPU uncorrectable read error occurred.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MUECLR register.
1	UDMAWE		M3 µDMA Uncorrectable Write Error Status Flag.
		0	No M3 µDMA uncorrectable write error occurred.
		1	M3 µDMA uncorrectable write error occurred.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MUECLR register.
0	M3CPUWE		M3 CPU Uncorrectable Write Error Status Flag.
		0	No M3 CPU uncorrectable write error occurred.
		1	M3 CPU uncorrectable write error occurred.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MUECLR register.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.8 M3 Uncorrectable Error Force Register (MUEFRC)

Figure 5-23. M3 Uncorrectable Error Force Register (MUEFRC)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		UDMARE	M3CPURE	UDMAWE	M3CPUWE
R-0		R/W=1-0	R/W=1-0	R/W=1-0	R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-28. M3 Uncorrectable Error Force Register (MUEFRC) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reserved
3	UDMARE		M3 µDMA Uncorrectable Read Error Force .Any reads to this bit will return a 0. Setting this bit to 1 will set the M3 µDMA uncorrectable read error flag status.
2	M3CPURE		M3 CPU Uncorrectable Read Error Force. Any reads to this bit will return a 0. Setting this bit to 1 will set the M3 CPU uncorrectable read error flag status.
1	UDMAWE		M3 µDMA Uncorrectable Write Error Force. Any reads to this bit will return a 0. Setting this bit to 1 will set the M3 µDMA uncorrectable write error flag status.
0	M3CPUWE		M3 CPU Uncorrectable Write Error Force. Any reads to this bit will return a 0. Setting this bit to 1 will set the M3 CPU uncorrectable write error flag status.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.9 M3 Uncorrectable Error Flag Clear Register (MUECLR)

Figure 5-24. M3 Uncorrectable Error Flag Clear Register (MUECLR)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		UDMARE	M3CPURE	UDMAWE	M3CPUWE
R-0		R/W=1-0	R/W=1-0	R/W=1-0	R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-29. M3 Uncorrectable Error Flag Clear Register (MUECLR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reserved
3	UDMARE		M3 µDMA Uncorrectable Read Error Clear. Any reads to this bit will return a 0.
		0	No effect
		1	Clears the M3 µDMA uncorrectable read error flag.
2	M3CPURE		M3 CPU Uncorrectable Read Error Clear. Any reads to this bit will return a 0.
		0	No effect
		1	Clears the M3 CPU uncorrectable read error flag.
1	UDMAWE		M3 µDMA Uncorrectable Write Error Clear. Any reads to this bit will return a 0.
		0	No effect
		1	Clears the M3 µDMA uncorrectable write error flag.
0	M3CPUWE		M3 CPU Uncorrectable Write Error Clear. Any reads to this bit will return a 0.
		0	No effect
		1	Clears the M3 CPU uncorrectable write error flag.

5.2.2.10 M3 Corrected Error Counter Register (MCECNTR)

Figure 5-25. M3 Corrected Error Counter Register (MCECNTR)

31	16	15	0
Reserved		MCECNTR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-30. M3 Corrected Error Counter Register (MCECNTR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	MCECNTR		M3 CPU/µDMA Corrected Error Counter In case of an error that has been corrected during M3 CPU or µDMA reads, this counter increments by 1. After increment, if this counter value becomes equal to the value configured in the MCETRES register, correctable error interrupt gets generated if it is enableds in the MCEIE register. **Note:** Writing a value equal to the MCETRES generates an interrupt and sets the MCEFLG.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.11 M3 Corrected Error Threshold Register (MCETRES)

Figure 5-26. M3 Corrected Error Threshold Register (MCETRES)

31	16	15	0
Reserved		MCETRES	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-31. M3 Corrected Error Threshold Register (MCETRES) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	MCETRES		M3 CPU/µDMA Corrected Error Threshold Value If MCECNTR = MCETRES, correctable error interrupt gets generated if it is enableds in the MCEIE register.

5.2.2.12 M3 Corrected Error Threshold Exceeded Flag Register (MCEFLG)

Figure 5-27. M3 Corrected Error Threshold Exceeded Flag Register (MCEFLG)

31	1	0
Reserved		MCEFLG
R-0		M3 R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-32. M3 Corrected Error Threshold Exceeded Flag Register (MCEFLG) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	MCEFLG		M3 CPU/µDMA Corrected Error Count Reached Flag This status flag is set when corrected error count on M3 CPU or µDMA accesses becomes equal to the M3 CPU/µDMA corrected error threshold. Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MCECLR register.

5.2.2.13 M3 Corrected Error Threshold Exceeded Force Register (MCEFRC)

Figure 5-28. M3 Corrected Error Threshold Exceeded Force Register (MCEFRC)

31	1	0
Reserved		MCEFRC
R-0		R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-33. M3 Corrected Error Threshold Exceeded Force Register (MCEFRC) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	MCEFRC		M3 Correctable Error Flag Force. Any read to this bit returns a 0. Setting this bit to 1 sets the MCEFLG flag in the MCEFLG register.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.14 M3 Corrected Error Threshold Exceeded Flag Clear Register (MCECLR)

Figure 5-29. M3 Corrected Error Threshold Exceeded Flag Clear Register (MCECLR)

31	1	0
Reserved		MCECLR
R-0		R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-34. M3 Corrected Error Threshold Exceeded Flag Clear Register (MCECLR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	MCECLR		M3 Corrected Error Threshold Reached Error Flag Clear. Any reads to this bit will return a 0. Writing a 1 to this bit clears the M3 corrected error threshold reached flag. It will also clear the MCECNTR register.

5.2.2.15 M3 Single Error Interrupt Enable Register (MCEIE)

Figure 5-30. M3 Single Error Interrupt Enable Register (MCEIE)

31	1	0
Reserved		MCEIE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-35. M3 Single Error Interrupt Enable Register (MCEIE) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	MCEIE		M3 CPU/µDMA Correctable Error Interrupt Enable
		0	Correctable error interrupt is not generated even though the MCEFLG flag is set.
		1	Correctable error interrupt is generated when the MCEFLG flag is set.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.16 Non-Master Access Violation Flag Register (MNMAVFLG)

Figure 5-31. Non-Master Access Violation Flag Register (MNMAVFLG)

31				16
Reserved				
R-0				
15		**3**	**2**	**1**	**0**
Reserved			CPUWRITE	DMAWRITE	CPUFETCH
R-0			R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-36. Non-Master Access Violation Flag Register (MNMAVFLG) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	CPUWRITE		Non-Master CPU Write Access Violation Flag
		0	Non-master CPU write access violation did not occur.
		1	Non-master CPU write access violation has occurred. The M3 CPU tried to write into an Sx RAM block for which C28x subsystem is the master.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MNMAVCLR register.
1	DMAWRITE		Non-Master DMA Write Access Violation Flag
		0	Non-master µDMA write access violation did not occur.
		1	Non-master µDMA write access violation has occurred. The M3 DMA tried to write into an Sx RAM block for which C28x subsystem is the master. In this case, writes are ignored.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MNMAVCLR register.
0	CPUFETCH		Non-Master CPU Fetch Access Violation Flag
		0	Non-master CPU fetch access violation did not occur.
		1	Non-master CPU fetch access violation has occurred. The M3 CPU tried to fetch code from an Sx RAM block for which C28x subsystem is the master.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MNMAVCLR register.

5.2.2.17 Non-Master Access Violation Flag Clear Register (MNMAVCLR)

Figure 5-32. Non-Master Access Violation Flag Clear Register (MNMAVCLR)

31				16
Reserved				
R-0				
15		**3**	**2**	**1**	**0**
Reserved			CPUWRITE	DMAWRITE	CPUFETCH
R-0			R=0/W=1-0	R=0/W=1-0	R=0/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-37. Non-Master Access Violation Flag Clear Register (MNMAVCLR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	CPUWRITE		Non-Master CPU Write Access Violation Clear
		0	No effect.
		1	Clears the corresponding non-master DMA write access violation flag.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-37. Non-Master Access Violation Flag Clear Register (MNMAVCLR) Field Descriptions (continued)

Bit	Field	Value	Description
1	DMAWRITE		Non-Master DMA Write Access Violation Flag
		0	Non-master µDMA write access violation did not occur.
		1	Non-master µDMA write access violation has occurred. The M3 DMA tried to write into an Sx RAM block for which C28x subsystem is the master. In this case, writes are ignored.
0	CPUFETCH		Non-Master CPU Fetch Access Violation Flag
		0	Non-master CPU fetch access violation did not occur.
		1	Non-master CPU fetch access violation has occurred. The M3 CPU tried to fetch code from an Sx RAM block for which C28x subsystem is the master.

5.2.2.18 Master Access Violation Flag Register (MMAVFLG)

Figure 5-33. Master Access Violation Flag Register (MMAVFLG)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		CPUWRITE	DMAWRITE	CPUFETCH
R-0		R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-38. Master Access Violation Flag Register (MMAVFLG) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	CPUWRITE		Master CPU Write Access Violation Flag
		0	Master CPU write access violation did not occur.
		1	Master CPU write access violation has occurred. The M3 CPU tried to write into a RAM Block for which CPUWRPROT is set to 1. Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MNMAVCLR register.
1	DMAWRITE		Master DMA Write Access Violation Flag
		0	Master DMA write access violation did not occur.
		1	Master DMA write access violation has occurred. The M3 µDMA tried to write into a RAM Block for which DMAWRPROT is set to 1. In this case, writes are ignored. Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MNMAVCLR register.
0	CPUFETCH		Master CPU Fetch Access Violation Flag
		0	Master CPU fetch access violation did not occur.
		1	Master CPU fetch access violation has occurred. The M3 CPU tried to fetch code from a RAM Block for which FETCHPROT is set to 1. Once this bit is set, it can be cleared by setting the corresponding error clear bit in the MNMAVCLR register.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.19 Master Access Violation Flag Clear Register (MMAVCLR)

Figure 5-34. Master Access Violation Flag Clear Register (MMAVCLR)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		CPUWRITE	DMAWRITE	CPUFETCH
R-0		R=0/W=1-0	R=0/W=1-0	R=0/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-39. Master Access Violation Flag Clear Register (MMAVCLR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	CPUWRITE		Master CPU Write Access Violation Clear
		0	No effect.
		1	Clears the corresponding master CPU write access violation flag.
1	DMAWRITE		Master DMA Write Access Violation Clear
		0	No effect.
		1	Clears the corresponding master DMA write access violation flag.
0	CPUFETCH		Master CPU Fetch Access Violation Clear
		0	No effect.
		1	Clears the corresponding master CPU fetch access violation flag.

5.2.2.20 Non-Master CPU Write Access Violation Address Register (MNMWRAVADDR)

Figure 5-35. Non-Master CPU Write Access Violation Address Register (MNMWRAVADDR)

31	0
NMCPUWRAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-40. Non-Master CPU Write Access Violation Address Register (MNMWRAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	NMCPUWRAVADDR		Non-Master CPU Write Access Violation Address This holds the address at which M3 CPU attempted a write access and the non-master CPU write access violation occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.21 Non-Master DMA Write Access Violation Address Register (MNMDMAWRAVADDR)

Figure 5-36. Non-Master DMA Write Access Violation Address Register (MNMDMAWRAVADDR)

31	0
NMDMAWRAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-41. Non-Master DMA Write Access Violation Address Register (MNMDMAWRAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	NMDMAWRAVADDR		Non-Master DMA Write Access Violation Address This holds the address at which M3 µDMA attempted a write access and the non-master DMA write access violation occurred.

5.2.2.22 Non-Master CPU Fetch Access Violation Address Register (MNMFAVADDR)

Figure 5-37. Non-Master CPU Fetch Access Violation Address Register (MNMFAVADDR)

31	0
NMCPUFAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-42. Non-Master CPU Fetch Access Violation Address Register (MNMFAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	NMCPUFAVADDR		Non-Master CPU Fetch Access Violation Address This holds the address at which M3 CPU attempted a code fetch and the non-master CPU fetch access violation occurred.

5.2.2.23 Master CPU Write Access Violation Address Register (MMWRAVADDR)

Figure 5-38. Master CPU Write Access Violation Address Register (MMWRAVADDR)

31	0
MCPUWRAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-43. Master CPU Write Access Violation Address Register (MMWRAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MCPUWRAVADDR		Master CPU Write Access Violation Address This holds the address at which M3 CPU attempted a write access and the master CPU write access violation occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.2.24 Master DMA Write Access Violation Address Register (MMDMAWRAVADDR)

Figure 5-39. Master DMA Write Access Violation Address Register (MMDMAWRAVADDR)

31	0
NMDMAWRAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-44. Master DMA Write Access Violation Address Register (CMDMAWRAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MDMAWRAVADDR		Master DMA Write Access Violation Address This holds the address at which M3 µDMA attempted a write access and the master DMA write access violation occurred.

Figure 5-40. Master CPU Fetch Access Violation Address Register (MMFAVADDR)

31	0
MCPUFAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-45. Master CPU Fetch Access Violation Address Register (MMFAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MCPUFAVADDR		Master CPU Fetch Access Violation Address This holds the address at which M3 CPU attempted a code fetch and the master CPU fetch access violation occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.3 C28x RAM Configuration Registers

5.2.3.1 Lx DEDRAM Configuration Register 1 (LxDRCR1)

Figure 5-41. Lx DEDRAM Configuration Register 1 (LxDRCR1)

31				16
Reserved				
R-0				

15	11	10	9	8
Reserved		CPUWRPROTL1	Reserved	FETCHPROTL1
R-0		R/W-0	R-0	R/W-0

7	3	2	1	0
Reserved		CPUWRPROTL0	Reserved	FETCHPROTL0
R-0		R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-46. Lx DEDRAM Configuration Register 1 (LxDRCR1) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved		Reserved
10	CPUWRPROTL1		CPU Write Protection L1
		0	C28x CPU write allowed to L1 RAM block.
		1	C28x CPU write not allowed to L1 RAM block.
9	Reserved		Reserved
8	FETCHPROTL1		CPU Fetch Protection L1
		0	C28x CPU Fetch allowed from L1 RAM block.
		1	C28x CPU Fetch not allowed from L1 RAM block.
7-3	Reserved		Reserved
2	CPUWRPROTL0		CPU Write Protection L0
		0	C28x CPU write allowed to L0 RAM block.
		1	C28x CPU write not allowed to L0 RAM block.
1	Reserved		Reserved
0	FETCHPROTL0		CPU Fetch Protection L0
		0	C28x CPU Fetch allowed from L0 RAM block.
		1	C28x CPU Fetch not allowed from L0 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.3.2 Lx SHRAM Configuration Register 1 (LxSRCR1)

Figure 5-42. Lx SHRAM Configuration Register 1 (LxSRCR1)

31	16
Reserved	
R-0	

15	11	10	9	8
Reserved		CPUWRPROTL3	DMAWRPROTL3	FETCHPROTL3
R-0		R/W-0	R/W-0	R/W-0

7	3	2	1	0
Reserved		CPUWRPROTC2	DMAWRPROTC2	FETCHPROTC2
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-47. Lx SHRAM Configuration Register 1 (LxSRCR1) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved		Reserved
10	CPUWRPROTL3		CPU Write Protection L3
		0	C28x CPU write allowed to L3 RAM block.
		1	C28x CPU write not allowed to L3 RAM block.
9	DMAWRPROTL3		DMA Write Protection L3
		0	C28x DMA write allowed to L3 RAM block.
		1	C28x DMA write not allowed to L3 RAM block.
8	FETCHPROTL3		CPU Fetch Protection L3
		0	C28x CPU Fetch allowed from L3 RAM block.
		1	C28x CPU Fetch not allowed from L3 RAM block.
7-3	Reserved		Reserved
2	CPUWRPROTC2		CPU Write Protection L2
		0	C28x CPU write allowed to L2 RAM block.
		1	C28x CPU write not allowed to L2 RAM block.
1	DMAWRPROTC2		DMA Write Protection L2
		0	C28x DMA write allowed to L2 RAM block.
		1	C28x DMA write not allowed to L2 RAM block.
0	FETCHPROTC2		CPU Fetch Protection L2
		0	C28x CPU Fetch allowed from L2 RAM block.
		1	C28x CPU Fetch not allowed from L2 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.3.3 C28x Sx SHRAM Master Select Register (CSxMSEL)

Figure 5-43. C28x Sx SHRAM Master Select Register (CSxMSEL)

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

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
S7MSEL	S6MSEL	S5MSEL	S4MSEL	S3MSEL	S2MSEL	S1MSEL	S0MSEL
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-48. C28x Sx SHRAM Master Select Register (CSxMSEL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7	S7MSEL		Master Ownership for S7 RAM Block
		0	M3 subsystem is master for S7 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S7 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
6	S6MSEL		Master Ownership for S6 RAM Block
		0	M3 subsystem is master for S6 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S6 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
5	S5MSEL		Master Ownership for S5 RAM Block
		0	M3 subsystem is master for S5 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S5 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
4	S4MSEL		Master Ownership for S4 RAM Block
		0	M3 subsystem is master for S4 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S4 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
3	S3MSEL		Master Ownership for S3 RAM Block
		0	M3 subsystem is master for S3 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S3 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
2	S2MSEL		Master Ownership for S2 RAM Block
		0	M3 subsystem is master for S2 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S2 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.
1	S1MSEL		Master Ownership for S1 RAM Block
		0	M3 subsystem is master for S1 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S1 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-48. C28x Sx SHRAM Master Select Register (CSxMSEL) Field Descriptions (continued)

Bit	Field	Value	Description
0	S0MSEL		Master Ownership for S0 RAM Block
		0	M3 subsystem is master for S0 RAM block. M3 CPU/µDMA accesses are allowed based on the setting of protection bits in the MSxSRCR register.
		1	C28 subsystem is master for S0 RAM block. C28 CPU/DMA accesses are allowed based on the setting of protection bits in the CSxSRCR register.

5.2.3.4 C28x Sx SHRAM Configuration Register 1 (CSxSRCR1)

Figure 5-44. C28x Sx SHRAM Configuration Register 1 (CSxSRCR1)

31	27	26	25	24
Reserved		CPUWRPROTS3	DMAWRPROTS3	FETCHPROTS3
R-0		R/W-0	R/W-0	R/W-0
23	19	18	17	16
Reserved		CPUWRPROTS2	DMAWRPROTS2	FETCHPROTS2
R-0		R/W-0	R/W-0	R/W-0
15	11	10	9	8
Reserved		CPUWRPROTS1	DMAWRPROTS1	FETCHPROTS1
R-0		R/W-0	R/W-0	R/W-0
7	3	2	1	0
Reserved		CPUWRPROTS0	DMAWRPROTS0	FETCHPROTS0
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-49. C28x Sx SHRAM Configuration Register 1 (CSxSRCR1) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26	CPUWRPROTS3		CPU Write Protection S3
		0	C28x CPU write allowed to S3 RAM block.
		1	C28x CPU write not allowed to S3 RAM block.
25	DMAWRPROTS3		DMA Write Protection S3
		0	C28x DMA write allowed to S3 RAM block.
		1	C28x DMA write not allowed to S3 RAM block.
24	FETCHPROTS3		CPU Fetch Protection S3
		0	C28x CPU Fetch allowed from S3 RAM block.
		1	C28x CPU Fetch not allowed from S3 RAM block.
23-19	Reserved		Reserved
18	CPUWRPROTS2		CPU Write Protection S2
		0	C28x CPU write allowed to S2 RAM block.
		1	C28x CPU write not allowed to S2 RAM block.
17	DMAWRPROTS2		DMA Write Protection S2
		0	C28x DMA write allowed to S2 RAM block.
		1	C28x DMA write not allowed to S2 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-49. C28x Sx SHRAM Configuration Register 1 (CSxSRCR1) Field Descriptions (continued)

Bit	Field	Value	Description
16	FETCHPROTS2		CPU Fetch Protection S2
		0	C28x CPU Fetch allowed from S2 RAM block.
		1	C28x CPU Fetch not allowed from S2 RAM block.
15-11	Reserved		Reserved
10	CPUWRPROTS1		CPU Write Protection S1
		0	C28x CPU write allowed to S1 RAM block.
		1	C28x CPU write not allowed to S1 RAM block.
9	DMAWRPROTS1		DMA Write Protection S1
		0	C28x DMA write allowed to S1 RAM block.
		1	C28x DMA write not allowed to S1 RAM block.
8	FETCHPROTS1		CPU Fetch Protection S1
		0	C28x CPU Fetch allowed from S1 RAM block.
		1	C28x CPU Fetch not allowed from S1 RAM block.
7-3	Reserved		Reserved
2	CPUWRPROTS0		CPU Write Protection S0
		0	C28x CPU write allowed to S0 RAM block.
		1	C28x CPU write not allowed to S0 RAM block.
1	DMAWRPROTS0		DMA Write Protection S0
		0	C28x DMA write allowed to S0 RAM block.
		1	C28x DMA write not allowed to S0 RAM block.
0	FETCHPROTS0		CPU Fetch Protection S0
		0	C28x CPU Fetch allowed from S0 RAM block.
		1	C28x CPU Fetch not allowed from S0 RAM block.

5.2.3.5 C28x Sx SHRAM Configuration Register 2 (CSxSRCR2)

Figure 5-45. C28x Sx SHRAM Configuration Register 2 (CSxSRCR2)

31	27	26	25	24
Reserved		CPUWRPROTS7	DMAWRPROTS7	FETCHPROTS7
R-0		R/W-0	R/W-0	R/W-0
23	19	18	17	16
Reserved		CPUWRPROTS6	DMAWRPROTS6	FETCHPROTS6
R-0		R/W-0	R/W-0	R/W-0
15	11	10	9	8
Reserved		CPUWRPROTS5	DMAWRPROTS5	FETCHPROTS5
R-0		R/W-0	R/W-0	R/W-0
7	3	2	1	0
Reserved		CPUWRPROTS4	DMAWRPROTS4	FETCHPROTS4
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-50. C28x Sx SHRAM Configuration Register 2 (CSxSRCR2) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26	CPUWRPROTS7		CPU Write Protection S7
		0	C28x CPU write allowed to S7 RAM block.
		1	C28x CPU write not allowed to S7 RAM block.
25	DMAWRPROTS7		DMA Write Protection S7
		0	C28x DMA write allowed to S7 RAM block.
		1	C28x DMA write not allowed to S7 RAM block.
24	FETCHPROTS7		CPU Fetch Protection S7
		0	C28x CPU Fetch allowed from S7 RAM block.
		1	C28x CPU Fetch not allowed from S7 RAM block.
23-19	Reserved		Reserved
18	CPUWRPROTS6		CPU Write Protection S6
		0	C28x CPU write allowed to S6 RAM block.
		1	C28x CPU write not allowed to S6 RAM block.
17	DMAWRPROTS6		DMA Write Protection S6
		0	C28x DMA write allowed to S6 RAM block.
		1	C28x DMA write not allowed to S6 RAM block.
16	FETCHPROTS6		CPU Fetch Protection S6
		0	C28x CPU Fetch allowed from S6 RAM block.
		1	C28x CPU Fetch not allowed from S6 RAM block.
15-11	Reserved		Reserved
10	CPUWRPROTS5		CPU Write Protection S5
		0	C28x CPU write allowed to S5 RAM block.
		1	C28x CPU write not allowed to S5 RAM block.
9	DMAWRPROTS5		DMA Write Protection S5
		0	C28x DMA write allowed to S5 RAM block.
		1	C28x DMA write not allowed to S5 RAM block.
8	FETCHPROTS5		CPU Fetch Protection S5
		0	C28x CPU Fetch allowed from S5 RAM block.
		1	C28x CPU Fetch not allowed from S5 RAM block.
7-3	Reserved		Reserved
2	CPUWRPROTS4		CPU Write Protection S4
		0	C28x CPU write allowed to S4 RAM block.
		1	C28x CPU write not allowed to S4 RAM block.
1	DMAWRPROTS4		DMA Write Protection S4
		0	C28x DMA write allowed to S4 RAM block.
		1	C28x DMA write not allowed to S4 RAM block.
0	FETCHPROTS4		CPU Fetch Protection S4
		0	C28x CPU Fetch allowed from S4 RAM block.
		1	C28x CPU Fetch not allowed from S4 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.3.6 C28TOC28_MSG_RAM Configuration Register (CTOMMSGRCR)

Figure 5-46. C28TOC28_MSG_RAM Configuration Register (CTOMMSGRCR)

31	2	1	0
Reserved		DMAWRPROT	Rsvd
R-0		R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-51. C28TOC28_MSG_RAM Configuration Register (CTOMMSGRCR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	DMAWRPROT		DMA Write Protection
		0	C28x DMA write allowed to CTOM_MSG_RAM.
		1	C28x DMA write not allowed to CTOM_MSG_RAM.
0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.3.7 M0, M1 and C28T0C28_MSG_RAM Test and Initialization Register (C28RTESTINIT)

Figure 5-47. M0, M1 and C28T0C28_MSG_RAM Test and Initialization Register (C28RTESTINIT)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved		ECCPARTESTCTOMMSGRAM	RAMINITCTOMMSGRAM	ECCPARTESTM1	RAMINITM1	ECCPARTESTM0	RAMINITM0
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-52. M0, M1 and C28T0C28_MSG_RAM Test and Initialization Register (C28RTESTINIT) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5	ECCPARTESTCTOMMSGRAM		Enable/Disable RAMTEST Feature for CTOM_MSG_RAM
		0	RAMTEST feature is disabled for CTOM_MSG_RAM block.
		1	RAMTEST feature is enabled for CTOM_MSG_RAM block. ECC/parity logic is bypassed for memory accesses.
4	RAMINITCTOMMSGRAM		RAM Initialization for CTOM_MSG_RAM Block. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of CTOM_MSG_RAM block with data 0x0 and corresponding data an address ECC/parity bits.
3	ECCPARTESTM1		Enable/Disable RAMTEST Feature for M1 RAM Block
		0	RAMTEST feature is disabled for M1 RAM block.
		1	RAMTEST feature is enabled for M1 RAM block. ECC/parity logic is bypassed for memory accesses.
2	RAMINITM1		RAM Initialization M1. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of M1 RAM block with data 0x0 and corresponding data an address ECC/parity bits.
1	ECCPARTESTM0		Enable/Disable RAMTEST Feature for M0 RAM Block
		0	RAMTEST feature is disabled for M0 RAM block.
		1	RAMTEST feature is enabled for M0 RAM block. ECC/parity logic is bypassed for memory accesses.
0	RAMINITM0		RAM Initialization M0. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of M0 RAM block with data 0x0 and corresponding data an address ECC/parity bits.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.3.8 Lx RAM Test and Initialization Register 1 (CLxRTESTINIT1)

Figure 5-48. Lx RAM Test and Initialization Register 1 (CLxRTESTINIT1)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
ECCPARTESTL3	RAMINITL3	ECCPARTESTL2	RAMINITL2	ECCPARTESTL1	RAMINITL1	ECCPARTESTL0	RAMINITL0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-53. Lx RAM Test and Initialization Register 1 (CLxRTESTINIT1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7	ECCPARTESTL3		Enable/Disable RAMTEST Feature for L3 RAM Block
		0	RAMTEST feature is disabled for L3 RAM block.
		1	RAMTEST feature is enabled for L3 RAM block. ECC/parity logic is bypassed for memory accesses.
6	RAMINITL3		RAM Initialization L3. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of L3 RAM block with data 0x0 and corresponding data an address ECC/parity bits.
5	ECCPARTESTL2		Enable/Disable RAMTEST Feature for L2 RAM Block
		0	RAMTEST feature is disabled for L2 RAM block.
		1	RAMTEST feature is enabled for L2 RAM block. ECC/parity logic is bypassed for memory accesses.
4	RAMINITL2		RAM Initialization L2. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of L2 RAM block with data 0x0 and corresponding data an address ECC/parity bits.
3	ECCPARTESTL1		Enable/Disable RAMTEST Feature for L1 RAM Block
		0	RAMTEST feature is disabled for L1 RAM block.
		1	RAMTEST feature is enabled for L1 RAM block. ECC/parity logic is bypassed for memory accesses.
2	RAMINITL1		RAM Initialization L1. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of L1 RAM block with data 0x0 and corresponding data an address ECC/parity bits.
1	ECCPARTESTL0		Enable/Disable RAMTEST Feature for L0 RAM Block
		0	RAMTEST feature is disabled for L0 RAM block.
		1	RAMTEST feature is enabled for L0 RAM block. ECC/parity logic is bypassed for memory accesses.
0	RAMINITL0		RAM Initialization L0. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of L0 RAM block with data 0x0 and corresponding data an address ECC/parity bits.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.3.9 C28x Sx RAM Test and Initialization Register 1 (CSxRTESTINIT1)

Figure 5-49. C28x Sx RAM Test and Initialization Register 1 (CSxRTESTINIT1)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
ECCPARTESTS7	RAMINITS7	ECCPARTESTS6	RAMINITS6	ECCPARTESTS5	RAMINITS5	ECCPARTESTS4	RAMINITS4
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
ECCPARTESTS3	RAMINITS3	ECCPARTESTS2	RAMINITS2	ECCPARTESTS1	RAMINITS1	ECCPARTESTS0	RAMINITS0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-54. C28x Sx RAM Test and Initialization Register 1 (CSxRTESTINIT1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	ECCPARTESTS7		Enable/Disable RAMTEST Feature for S7 RAM Block if C28X Subsystem is Master for S7 RAM Block
		0	RAMTEST feature is disabled for S7 RAM block.
		1	RAMTEST feature is enabled for S7 RAM block. ECC/parity logic is bypassed for memory accesses.
14	RAMINITS7		RAM Initialization S7. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S7 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicable only if C28x subsystem is master for S7 memory.
13	ECCPARTESTS6		Enable/Disable RAMTEST Feature for S6 RAM Block if C28X Subsystem is Master for S6 RAM Block
		0	RAMTEST feature is disabled for S6 RAM block.
		1	RAMTEST feature is enabled for S6 RAM block. ECC/parity logic is bypassed for memory accesses.
12	RAMINITS6		RAM Initialization S6. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S6 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicable only if C28x subsystem is master for S6 memory.
11	ECCPARTESTS5		Enable/Disable RAMTEST Feature for S6 RAM Block if C28X Subsystem is Master for S5 RAM Block
		0	RAMTEST feature is disabled for S6 RAM block.
		1	RAMTEST feature is enabled for S6 RAM block. ECC/parity logic is bypassed for memory accesses.
10	RAMINITS5		RAM Initialization S6. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S6 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicable only if C28x subsystem is master for S5 memory.
9	ECCPARTESTS4		Enable/Disable RAMTEST Feature for S4 RAM Block if C28X Subsystem is Master for S4 RAM Block
		0	RAMTEST feature is disabled for S4 RAM block.
		1	RAMTEST feature is enabled for S4 RAM block. ECC/parity logic is bypassed for memory accesses.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-54. C28x Sx RAM Test and Initialization Register 1 (CSxRTESTINIT1) Field Descriptions (continued)

Bit	Field	Value	Description
8	RAMINITS4		RAM Initialization S4. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S4 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicable only if C28x subsystem is master for S4 memory.
7	ECCPARTESTS3		Enable/Disable RAMTEST Feature for S3 RAM Block if C28X Subsystem is Master for S3 RAM Block
		0	RAMTEST feature is disabled for S3 RAM block.
		1	RAMTEST feature is enabled for S3 RAM block. ECC/parity logic is bypassed for memory accesses.
6	RAMINITS3		RAM Initialization S3. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S3 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicable only if C28x subsystem is master for S3 memory.
5	ECCPARTESTS2		Enable/Disable RAMTEST Feature for S2 RAM Block if C28X Subsystem is Master for S2 RAM Block
		0	RAMTEST feature is disabled for S2 RAM block.
		1	RAMTEST feature is enabled for S2 RAM block. ECC/parity logic is bypassed for memory accesses.
4	RAMINITS2		RAM Initialization S2. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S2 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicable only if C28x subsystem is master for S2 memory.
3	ECCPARTESTS1		Enable/Disable RAMTEST Feature for S1 RAM Block if C28X Subsystem is Master for S1 RAM Block
		0	RAMTEST feature is disabled for S1 RAM block.
		1	RAMTEST feature is enabled for S1 RAM block. ECC/parity logic is bypassed for memory accesses.
2	RAMINITS1		RAM Initialization S1. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S1 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicable only if C28x subsystem is master for S1 memory.
1	ECCPARTESTS0		Enable/Disable RAMTEST Feature for S0 RAM Block if C28X Subsystem is Master for S0 RAM Block
		0	RAMTEST feature is disabled for S0 RAM block.
		1	RAMTEST feature is enabled for S0 RAM block. ECC/parity logic is bypassed for memory accesses.
0	RAMINITS0		RAM Initialization S0. Any reads to this bit will return a 0.
		0	No action taken.
		1	Initialize all address locations of S0 RAM block with data 0x0 and corresponding data an address ECC/parity bits. Applicable only if C28x subsystem is master for S0 memory.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.3.10 M0, M1 and C28T0M3_MSG_RAM INIT Done Register (C28RINITDONE)

Figure 5-50. M0, M1 and C28T0M3_MSG_RAM INIT Done Register (C28RINITDONE)

31	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		RAMINITDONECTOMMSGRAM	Reserved	RAMINITDONEM1	Reserved	RAMINITDONEM0
R-0		R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-55. M0, M1 and C28T0M3_MSG_RAM INIT Done Register (C28RINITDONE) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4	RAMINITDONECTO MMSGRAM		RAM Initialization Process Status when RAMINIT is Set for CTOM_MSG_RAM Block
			RAM initialization is not finished for CTOM_MSG_RAM block.
		0	RAM initialization is done for CTOM_MSG_RAM block. CTOM_MSG_RAM can be accessed by M3 CPU/µDMA.
		1	This status bit gets cleared when the RAMINIT bit is set for C0 RAM block.
3	Reserved		Reserved
2	RAMINITDONEM1		RAM Initialization Process Status when RAMINIT is Set for M1 RAM Block
			RAM initialization is not finished for M1 RAM block.
		0	RAM initialization is done for M1 RAM block. M1 RAM can be accessed by M3 CPU.
		1	This status bit gets cleared when the RAMINIT bit is set for M1 RAM block.
1	Reserved		Reserved
0	RAMINITDONEM0		RAM Initialization Process Status when RAMINIT is Set for M0 RAM Block
			RAM initialization is not finished for M0 RAM block.
		0	RAM initialization is done for M0 RAM block. M0 RAM can be accessed by M3 CPU.
		1	This status bit gets cleared when the RAMINIT bit is set for M0 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.3.11 C28x Lx RAM_INIT_DONE Register 1 (CLxRINITDONE1)

Figure 5-51. C28x Lx RAM_INIT_DONE Register 1 (CLxRINITDONE1)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved	RAMINITDONEC3	Reserved	RAMINITDONEC2	Reserved	RAMINITDONEC1	Reserved	RAMINITDONEC0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-56. C28x Lx RAM_INIT_DONE Register 1 (CLxRINITDONE1) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved		Reserved
6	RAMINITDONELC3		RAM Initialization Process Status when RAMINIT is Set for C3 RAM Block
		0	RAM initialization is not finished for C3 RAM block.
		1	RAM initialization is done for C3 RAM block. C3 RAM can be accessed by M3 CPU.
			This status bit gets cleared when the RAMINIT bit is set for C3 RAM block.
5	Reserved		Reserved
4	RAMINITDONEC2		RAM Initialization Process Status when RAMINIT is Set for C2 RAM Block
			RAM initialization is not finished for C2 RAM block.
		0	RAM initialization is done for C2 RAM block. C2 RAM can be accessed by M3 CPU.
		1	This status bit gets cleared when the RAMINIT bit is set for C2 RAM block.
3	Reserved		Reserved
2	RAMINITDONEC1		RAM Initialization Process Status when RAMINIT is Set for C1 RAM Block
			RAM initialization is not finished for C1 RAM block.
		0	RAM initialization is done for C1 RAM block. C1 RAM can be accessed by M3 CPU.
		1	This status bit gets cleared when the RAMINIT bit is set for C1 RAM block.
1	Reserved		Reserved
0	RAMINITDONEC0		RAM Initialization Process Status when RAMINIT is Set for C0 RAM Block
			RAM initialization is not finished for C0 RAM block.
		0	RAM initialization is done for C0 RAM block. C0 RAM can be accessed by M3 CPU.
		1	This status bit gets cleared when the RAMINIT bit is set for C0 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.3.12 C28x Sx RAM_INIT_DONE Register 1 (CSxRINITDONE1)

Figure 5-52. C28x Sx RAM_INIT_DONE Register 1 (CSxRINITDONE1)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	RAMINITDONES7	Reserved	RAMINITDONES6	Reserved	RAMINITDONES5	Reserved	RAMINITDONES4
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
Reserved	RAMINITDONES3	Reserved	RAMINITDONES2	Reserved	RAMINITDONES1	Reserved	RAMINITDONES0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-57. C28x Sx RAM_INIT_DONE Register 1 (CSxRINITDONE1) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reserved
14	RAMINITDONES7		RAM Initialization Process Status when RAMINIT is Set for S7 RAM Block
		0	RAM initialization is not finished for S7 RAM block.
		1	RAM initialization is done for S7 RAM block. S7 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA. This status bit gets cleared when the RAMINIT bit is set for S7 RAM block.
13	Reserved		Reserved
12	RAMINITDONES6		RAM Initialization Process Status when RAMINIT is Set for S6 RAM Block
		0	RAM initialization is not finished for S6 RAM block.
		1	RAM initialization is done for S6 RAM block. S6 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA. This status bit gets cleared when the RAMINIT bit is set for S6 RAM block.
11	Reserved		Reserved
10	RAMINITDONES5		RAM Initialization Process Status when RAMINIT is Set for S5 RAM Block
		0	RAM initialization is not finished for S5 RAM block.
		1	RAM initialization is done for S7\5 RAM block. S5 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA. This status bit gets cleared when the RAMINIT bit is set for S5 RAM block.
9	Reserved		Reserved
8	RAMINITDONES4		RAM Initialization Process Status when RAMINIT is Set for S4 RAM Block
		0	RAM initialization is not finished for S4 RAM block.
		1	RAM initialization is done for S4 RAM block. S4 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA. This status bit gets cleared when the RAMINIT bit is set for S4 RAM block.
7	Reserved		Reserved
6	RAMINITDONES3		RAM Initialization Process Status when RAMINIT is Set for S3 RAM Block
		0	RAM initialization is not finished for S3 RAM block.
		1	RAM initialization is done for S3 RAM block. S3 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA. This status bit gets cleared when the RAMINIT bit is set for S3 RAM block.
5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-57. C28x Sx RAM_INIT_DONE Register 1 (CSxRINITDONE1) Field Descriptions (continued)

Bit	Field	Value	Description
4	RAMINITDONES2		RAM Initialization Process Status when RAMINIT is Set for S2 RAM Block
		0	RAM initialization is not finished for S2 RAM block.
		1	RAM initialization is done for S2 RAM block. S2 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S2 RAM block.
3	Reserved		Reserved
2	RAMINITDONES1		RAM Initialization Process Status when RAMINIT is Set for S1 RAM Block
		0	RAM initialization is not finished for S1 RAM block.
		1	RAM initialization is done for S1 RAM block. S1 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S1 RAM block.
1	Reserved		Reserved
0	RAMINITDONES0		RAM Initialization Process Status when RAMINIT is Set for S0 RAM Block
		0	RAM initialization is not finished for S0 RAM block.
		1	RAM initialization is done for S0 RAM block. S0 RAM can be accessed by M3 CPU/µDMA or C28x CPU/DMA.
			This status bit gets cleared when the RAMINIT bit is set for S0 RAM block.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4 C28x RAM Error Registers

5.2.4.1 C28x CPU Uncorrectable Read Error Address Register (CCUNCREADDR)

Figure 5-53. C28x CPU Uncorrectable Read Error Address Register (CCUNCREADDR)

31	0
CCUNCREADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-58. C28x CPU Uncorrectable Read Error Address Register (CCUNCREADDR) Field Descriptions

Bit	Field	Value	Description
31-0	CCUNCREADDR		This register contains the address where uncorrectable error occurs during C28x CPU data read or fetch. Only the address coresponding to the last error is stored.

5.2.4.2 C28x DMA Uncorrectable Read Error Address Register (CDUNCREADDR)

Figure 5-54. C28x DMA Uncorrectable Read Error Address Register (CDUNCREADDR)

31	0
CDUNCREADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-59. C28x DMA Uncorrectable Read Error Address Register (CDUNCREADDR) Field Descriptions

Bit	Field	Value	Description
31-0	CDUNCREADDR		This register contains the address where uncorrectable error occurs during C28x DMA data read. Only the address coresponding to the last error is stored.

5.2.4.3 C28x CPU Corrected Read Error Address Register (CCPUCREADDR)

Figure 5-55. C28x CPU Corrected Read Error Address Register (CCPUCREADDR)

31	0
CCPUCREADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-60. C28x CPU Corrected Read Error Address Register (CCPUCREADDR) Field Descriptions

Bit	Field	Value	Description
31-0	CCPUCREADDR		This register contains the address where correctable error occurs during C28x CPU data read or fetch. Only the address coresponding to the last error is stored.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.4 C28x DMA Corrected Read Error Address Register (CDMACREADDR)

Figure 5-56. C28x DMA Corrected Read Error Address Register (CDMACREADDR)

31	0
CDMACREADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-61. C28x DMA Corrected Read Error Address Register (CDMACREADDR) Field Descriptions

Bit	Field	Value	Description
31-0	CDMACREADDR		This register contains the address where correctable error occurs during C28x DMA data read. Only the address coresponding to the last error is stored.

5.2.4.5 C28x Uncorrectable Error Flag Register (CUEFLG)

Figure 5-57. C28x Uncorrectable Error Flag Register (CUEFLG)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		C28DMARE	C28CPURE
R-0		R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-62. C28x Uncorrectable Error Flag Register (CUEFLG) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	C28DMARE		C28x DMA Uncorrectable Read Error Status Flag
		0	No C28x DMA uncorrectable read error occurred.
		1	C28x DMA uncorrectable read error occurred.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the CUECLR register.
0	C28CPURE		C28x CPU Uncorrectable Read Error Status Flag
		0	No C28x CPU uncorrectable read error occurred.
		1	C28x CPU uncorrectable read error occurred.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the CUECLR register.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.6 C28x Uncorrectable Error Force Register (CUEFRC)

Figure 5-58. C28x Uncorrectable Error Force Register (CUEFRC)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		DMARE	C28CPURE
R-0		R/W=1-0	R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-63. C28x Uncorrectable Error Force Register (CUEFRC) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	DMARE		C28x DMA Uncorrectable Read Error Force. Any reads to this bit will return a 0. Setting this bit to 1 will set the C28x DMA uncorrectable read error flag status.
0	C28CPURE		C28x CPU Uncorrectable Read Error Force. Any reads to this bit will return a 0. Setting this bit to 1 will set the C28x CPU uncorrectable read error flag status.

5.2.4.7 C28x Uncorrectable Error Flag Clear Register (CUECLR)

Figure 5-59. C28x Uncorrectable Error Flag Clear Register (CUECLR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		DMARE	C28CPURE
R-0		R/W=1-0	R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-64. C28x Uncorrectable Error Flag Clear Register (CUECLR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reserved
1	DMARE		C28x DMA Uncorrectable Read Error Clear. Any reads to this bit will return a 0.
		0	No effect
		1	Clears the C28x DMA uncorrectable read error flag.
0	C28CPURE		C28x CPU Uncorrectable Read Error Clear. Any reads to this bit will return a 0.
		0	No effect
		1	Clears the C28x CPU uncorrectable read error flag.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.8 C28x Corrected Error Counter Register (CCECNTR)

Figure 5-60. C28x Corrected Error Counter Register (CCECNTR)

31	16	15	0
Reserved		CCECNTR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-65. C28x Corrected Error Counter Register (CCECNTR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	CCECNTR		C28x CPU/DMA Corrected Error Counter In case of an error that has been corrected during C28x CPU or DMA reads, this counter increments by 1. After increment, if this counter value becomes equal to the value configured in the CCETRES register, correctable error interrupt gets generated if it is enabled in the CCEIE register. **Note:** Writing a value equal to the CCETRES generates an interrupt and sets the CCEFLG.

5.2.4.9 C28x Corrected Error Threshold Register (CCETRES)

Figure 5-61. C28x Corrected Error Threshold Register (CCETRES)

31	16	15	0
Reserved		CCETRES	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-66. C28x Corrected Error Threshold Register (CCETRES) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	CCETRES		C28x CPU/DMA Corrected Error Threshold Value If CCECNTR = CCETRES, correctable error interrupt gets generated if it is enabled in the CCEIE register.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.10 C28x Corrected Error Threshold Exceeded Flag Register (CCEFLG)

Figure 5-62. C28x Corrected Error Threshold Exceeded Flag Register (CCEFLG)

31	1	0
Reserved		CCEFLG
R-0		C28x R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-67. C28x Corrected Error Threshold Exceeded Flag Register (CCEFLG) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	CCEFLG		C28x CPU/DMA Corrected Error Count Reached Flag This status flag is set when corrected error count on C28x CPU or DMA accesses becomes equal to the C28x CPU/DMA corrected error threshold. Once this bit is set, it can be cleared by setting the corresponding error clear bit in the CCECLR register.

5.2.4.11 C28x Corrected Error Threshold Exceeded Force Register (CCEFRC)

Figure 5-63. C28x Corrected Error Threshold Exceeded Force Register (CCEFRC)

31	1	0
Reserved		CCEFRC
R-0		R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-68. C28x Corrected Error Threshold Exceeded Force Register (CCEFRC) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	CCEFRC		C28x Correctable Error Flag Force. Any reads to this bit will return a 0. Setting this bit to 1 sets the CCEFLG flag in the CCEFLG register.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.12 C28x Corrected Error Threshold Exceeded Flag Clear Register (CCECLR)

Figure 5-64. C28x Corrected Error Threshold Exceeded Flag Clear Register (CCECLR)

31	1	0
Reserved		CCECLR
R-0		R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-69. C28x Corrected Error Threshold Exceeded Flag Clear Register (CCECLR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	CCECLR		C28x Corrected Error Threshold Reached Error Flag Clear. Any reads to this bit will return a 0. Writing a 1 to this bit clears the C28x corrected error threshold reached flag. It will also clear the CCECNTR register.

5.2.4.13 C28x Single Error Interrupt Enable Register (CCEIE)

Figure 5-65. C28x Single Error Interrupt Enable Register (CCEIE)

31	1	0
Reserved		CCEIE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-70. C28x Single Error Interrupt Enable Register (CCEIE) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	CCEIE		C28x CPU/DMA Correctable Error Interrupt Enable
		0	Correctable error interrupt is not generated even though the CCEFLG flag is set.
		1	Correctable error interrupt is generated when the CCEFLG flag is set.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.14 Non-Master Access Violation Flag Register (CNMAVFLG)

Figure 5-66. Non-Master Access Violation Flag Register (CNMAVFLG)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		CPUWRITE	DMAWRITE	CPUFETCH
R-0		R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-71. Non-Master Access Violation Flag Register (CNMAVFLG) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	CPUWRITE		Non-Master CPU Write Access Violation Flag
		0	Non-master CPU write access violation did not occur.
		1	Non-master CPU write access violation has occurred. The C28x CPU tried to write into an Sx RAM block for which M3 subsystem is the master. Once this bit is set, it can be cleared by setting the corresponding error clear bit in the CNMAVCLR register.
1	DMAWRITE		Non-Master DMA Write Access Violation Flag
		0	Non-master DMA write access violation did not occur.
		1	Non-master DMA write access violation has occurred. The C28x DMA tried to write into an Sx RAM block for which M3 subsystem is the master. In this case, writes are ignored. Once this bit is set, it can be cleared by setting the corresponding error clear bit in the CNMAVCLR register.
0	CPUFETCH		Non-Master CPU Fetch Access Violation Flag
		0	Non-master CPU fetch access violation did not occur.
		1	Non-master CPU fetch access violation has occurred. The C28x CPU tried to fetch code from an Sx RAM block for which M3 subsystem is the master. Once this bit is set, it can be cleared by setting the corresponding error clear bit in the CNMAVCLR register.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.15 Non-Master Access Violation Force Register (CNMAVFRC)

Figure 5-67. Non-Master Access Violation Force Register (CNMAVFRC)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		CPUWRITE	DMAWRITE	CPUFETCH
R-0		R/W=1-0	R/W=1-0	R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-72. Non-Master Access Violation Force Register (CNMAVFRC) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	CPUWRITE		Non-Master CPU Write Access Violation Force. Any reads to this bit will return a 0.
		0	No effect.
		1	Sets the CPUFETCH flag in the CNMAVFLG register.
1	DMAWRITE		Non-Master DMA Write Access Violation Force. Any reads to this bit will return a 0.
		0	No effect.
		1	Sets the DMAWRITE flag in the CNMAVFLG register.
0	CPUFETCH		Non-Master CPU Fetch Access Violation Force. Any reads to this bit will return a 0.
		0	No effect.
		1	Sets the CPUFETCH flag in the CNMAVFLG register.

5.2.4.16 Non-Master Access Violation Flag Clear Register (CNMAVCLR)

Figure 5-68. Non-Master Access Violation Flag Clear Register (CNMAVCLR)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		CPUWRITE	DMAWRITE	CPUFETCH
R-0		R/W=1-0	R/W=1-0	R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-73. Non-Master Access Violation Flag Clear Register (CNMAVCLR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	CPUWRITE		Non-Master CPU Write Access Violation Clear. Any reads to this bit will return a 0.
		0	No effect.
		1	Clears the corresponding non-master CPU write access violation flag.
1	DMAWRITE		Non-Master DMA Write Access Violation Clear. Any reads to this bit will return a 0.
		0	No effect.
		1	Clears the corresponding non-master DMA write access violation flag.
0	CPUFETCH		Non-Master CPU Fetch Access Violation Clear. Any reads to this bit will return a 0.
		0	No effect.
		1	Clears the corresponding non-master CPU fetch access violation flag.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.17 Master Access Violation Flag Register (CMAVFLG)

Figure 5-69. Master Access Violation Flag Register (CMAVFLG)

31				16
Reserved				
R-0				
15		**2**	**1**	**0**
Reserved		CPUWRITE	DMAWRITE	CPUFETCH
R-0		R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 5-74. Master Access Violation Flag Register (CMAVFLG) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	CPUWRITE		Master CPU Write Access Violation Flag
		0	Master CPU write access violation did not occur.
		1	Master CPU write access violation has occurred. The C28x CPU tried to write into a RAM Block for which CPUWRPROT is set to 1.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the CNMAVCLR register.
1	DMAWRITE		Master DMA Write Access Violation Flag
		0	Master DMA write access violation did not occur.
		1	Master DMA write access violation has occurred. The C28x µDMA tried to write into a RAM Block for which DMAWRPROT is set to 1. In this case, writes are ignored.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the CNMAVCLR register.
0	CPUFETCH		Master CPU Fetch Access Violation Flag
		0	Master CPU fetch access violation did not occur.
		1	Master CPU fetch access violation has occurred. The C28x CPU tried to fetch code from a RAM Block for which FETCHPROT is set to 1.
			Once this bit is set, it can be cleared by setting the corresponding error clear bit in the CNMAVCLR register.

5.2.4.18 Master Access Violation Force Register (CMAVFRC)

Figure 5-70. Master Access Violation Force Register (CMAVFRC)

31				16
Reserved				
R-0				
15		**2**	**1**	**0**
Reserved		CPUWRITE	DMAWRITE	CPUFETCH
R-0		R/W=1-0	R/W=1-0	R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 5-75. Master Access Violation Force Register (CMAVFRC) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	CPUWRITE		Master CPU Write Access Violation Force. Any reads to this bit will return a 0.
		0	No effect.
		1	Sets the CPUFETCH flag in the CNMAVFLG register.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-75. Master Access Violation Force Register (CMAVFRC) Field Descriptions (continued)

Bit	Field	Value	Description
1	DMAWRITE		Master DMA Write Access Violation Force. Any reads to this bit will return a 0.
		0	No effect.
		1	Sets the DMAWRITE flag in the CNMAVFLG register.
0	CPUFETCH		Master CPU Fetch Access Violation Force. Any reads to this bit will return a 0.
		0	No effect.
		1	Sets the CPUFETCH flag in the CNMAVFLG register.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.19 Master Access Violation Flag Clear Register (CMAVCLR)

Figure 5-71. Master Access Violation Flag Clear Register (CMAVCLR)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		CPUWRITE	DMAWRITE	CPUFETCH
R-0		R/W=1-0	R/W=1-0	R/W=1-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-76. Master Access Violation Flag Clear Register (CMAVCLR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	CPUWRITE		Master CPU Write Access Violation Clea. Any reads to this bit will return a 0.
		0	No effect.
		1	Clears the corresponding master CPU write access violation flag.
1	DMAWRITE		Master DMA Write Access Violation Clear. Any reads to this bit will return a 0.
		0	No effect.
		1	Clears the corresponding master DMA write access violation flag.
0	CPUFETCH		Master CPU Fetch Access Violation Clear. Any reads to this bit will return a 0.
		0	No effect.
		1	Clears the corresponding master CPU fetch access violation flag.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.20 Non-Master CPU Write Access Violation Address Register (CNMWRAVADDR)

Figure 5-72. Non-Master CPU Write Access Violation Address Register (CNMWRAVADDR)

31	0
NMCPUWRAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-77. Non-Master CPU Write Access Violation Address Register (CNMWRAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	NMCPUWRAVADDR		Non-Master CPU Write Access Violation Address This holds the address at which C28x CPU attempted a write access and the non-master CPU write access violation occurred.

5.2.4.21 Non-Master DMA Write Access Violation Address Register (CNMDMAWRAVADDR)

Figure 5-73. Non-Master DMA Write Access Violation Address Register (CNMDMAWRAVADDR)

31	0
NMDMAWRAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-78. Non-Master DMA Write Access Violation Address Register (CNMDMAWRAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	NMDMAWRAVADDR		Non-Master DMA Write Access Violation Address This holds the address at which C28x DMA attempted a write access and the non-master DMA write access violation occurred.

5.2.4.22 Non-Master CPU Fetch Access Violation Address Register (CNMFAVADDR)

Figure 5-74. Non-Master CPU Fetch Access Violation Address Register (CNMFAVADDR)

31	0
NMCPUFAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-79. Non-Master CPU Fetch Access Violation Address Register (CNMFAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	NMCPUFAVADDR		Non-Master CPU Fetch Access Violation Address This holds the address at which C28x CPU attempted a code fetch and the non-master CPU fetch access violation occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

5.2.4.23 Master CPU Write Access Violation Address Register (CMWRAVADDR)

Figure 5-75. Master CPU Write Access Violation Address Register (CMWRAVADDR)

31	0
MCPUWRAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-80. Master CPU Write Access Violation Address Register (CMWRAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MCPUWRAVADDR		Master CPU Write Access Violation Address This holds the address at which C28x CPU attempted a write access and the master CPU write access violation occurred.

5.2.4.24 Master DMA Write Access Violation Address Register (CMDMAWRAVADDR)

Figure 5-76. Master DMA Write Access Violation Address Register (CMDMAWRAVADDR)

31	0
NMDMAWRAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-81. Master DMA Write Access Violation Address Register (CMDMAWRAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MDMAWRAVADDR		Master DMA Write Access Violation Address This holds the address at which C28x DMA attempted a write access and the master DMA write access violation occurred.

5.2.4.25 Master CPU Fetch Access Violation Address Register (CMFAVADDR)

Figure 5-77. Master CPU Fetch Access Violation Address Register (CMFAVADDR)

31	0
MCPUFAVADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-82. Master CPU Fetch Access Violation Address Register (CMFAVADDR) Field Descriptions

Bit	Field	Value	Description
31-0	MCPUFAVADDR		Master CPU Fetch Access Violation Address This holds the address at which C28x CPU attempted a code fetch and the master CPU fetch access violation occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

5.3 Flash Controller Memory Module

Flash is an electrically erasable/programmable nonvolatile memory that can be electrically programmed and erased many times to ease code development. Flash memory can be used primarily as a program memory for the core, and secondarily as static data memory.

This section describes the proper sequence to configure the wait states and operating mode of flash. It also includes information on flash and OTP power modes, how to improve flash performance by enabling the flash cache/prefetch mode, and the SECDED safety feature.

5.3.1 Features

Features of flash memory include:

- Dedicated flash bank in the master subsystem (refer to the device data manual for the size of flash bank)
- Dedicated flash bank in the control subsystem (refer to the device data manual for the size of flash bank)
- Multiple sectors providing the option of leaving some sectors programmed and only erasing specific sectors
- User-programmable OTP locations in the master subsystem
- Single flash pump shared by the master subsystem and control subsystem
- Hardware flash pump semaphore to control ownership of the pump between the two subsystems. The semaphore mechanism is not applicable during reads or program execution. It is also transparent when the CCS Plugin/UniFlash is used for flash programming. However, the semaphore must be managed when the API is used.
- Dedicated flash module controller (FMC) in the master subsystem and the control subsystem
- 128 bits (bankwidth) can be programmed at a time along with ECC
- Enhanced performance using program cache/code-prefetch mechanism and data cache in M3-FMC
- Enhanced performance using code-prefetch mechanism and data cache in C28x-FMC
- Configurable wait states to give the best performance for a given execution speed
- Safety Features
  - SECDED-single error correction and double error detection is supported in both FMCs
  - Address bits are included in ECC
  - Test mode to check the health of ECC logic
- Supports low-power modes for flash bank and pump for power savings
- Built-in power mode control logic
- Integrated program/erase state machine (FSM) in both FMCs
  - Simple flash API algorithms
  - Fast erase and program times (refer to the device data manual for details).
- Code Security Module to prevent access to the flash by unauthorized persons (refer to the *System Control and Interrupts* chapter for details)

5.3.2 Flash Tools

Texas Instruments provides the following tools for flash:

- Code Composer Studio V6.x - the development environment with integrated flash plugin
- F021 Flash API Library - a set of software functions to erase/program flash
  Please refer to the *C2000 F021 Flash Application Programming Interface (API) Reference Guide*, SPNU595 for more information.
- UniFlash - standalone tool to erase/program/verify the flash content through JTAG. No CCS is required.
- CCS On-Chip Flash Plugin and UniFlash tools are developed for a fully Flash-embedded application, which should be a standalone program that has all the initialized sections linked to flash. Any code that

Copyright © 2012–2019, Texas Instruments Incorporated

needs to be executed from RAM should be copied from Flash at run time.

- Users must check and install available updates for CCS On-Chip Flash Plugin and UniFlash tools.

5.3.3 Default Flash Configuration

The following are flash module configuration settings at power-up, in both master and control subsystems:

- Dedicated flash banks are in sleep power mode
- Shared pump is in sleep mode
- ECC is enabled
- Wait-states are set to max (0xF)
- Program cache and data cache are disabled in M3-FMC
- Code-prefetch mechanism and data cache are disabled in C28x-FMC

Note that during the boot process, the boot ROM performs a dummy read of the Code Security Module (CSM) password locations located in the flash. This read is performed to unlock a new (or erased) device that has no password stored in it, so that flash programming or loading of code into CSM protected SARAM can be performed. On devices with a password, this read has no effect and the device remains locked. One effect of this read is that the flash will transition from the sleep (reset) state to the active state.

User application software must initialize wait-states using the FRDCNTL register, and configure cache/prefetch features using the RD_INTF_CTRL register, to achieve optimum system performance. Software that configures flash settings like wait-states, cache/prefetch features, and so on, must be executed only from RAM memory, **not** from flash memory.

NOTE: Before initializing wait-states, cache/prefetch mechanisms must **always** be disabled, if they are enabled. After the initialization of wait-states is done, cache/prefetch mechanisms can be enabled as needed.

5.3.4 Flash Bank, OTP and Pump

There is a dedicated flash bank in the master subsystem called the M3 flash bank and a dedicated flash bank in the control subsystem called the C28x flash bank. Also, there is a one-time programmable (OTP) memory on the master subsystem called USER OTP which the user can program only once and cannot erase. Flash and OTP are uniformly mapped in both program and data memory space.

Both the master subsystem and control subsystem have a TI-OTP which contains manufacturing information like settings used by the flash state machine for erase and program operations, and so on. Users may read TI-OTP but cannot program or erase. For memory map and size information of the M3 flash bank, M3 TI-OTP, M3 User OTP, C28x flash bank, C28x TI-OTP and corresponding ECC locations, please refer to the device data manual.

The M3 flash bank/User OTP and C28x flash bank share a common flash pump. A hardware semaphore, called the flash pump semaphore, is provided to control the access of the flash pump between the master subsystem and control subsystem. Refer to the *System Control and Interrupts* chapter for more information on usage of flash pump semaphore

Table 5-83 provides the user-programmable OTP locations in M3 OTP. For more information on the functionality of these fields, please refer to the *System Control and Interrupts* chapter and the *ROM Code and Peripheral Booting* chapter.

Table 5-83. Programmable OTP Locations in M3 OTP

Name	Address	Size (x8)	Description
OTPSECLOCK	0x680800	4	Contents of this OTP location gets copied to the OTPSECLOCK register
Z2_FLASH_ENTRY_POINT	0x68080C	4	Alternate flash entry point can be programmed in this OTP location

Copyright © 2012–2019, Texas Instruments Incorporated

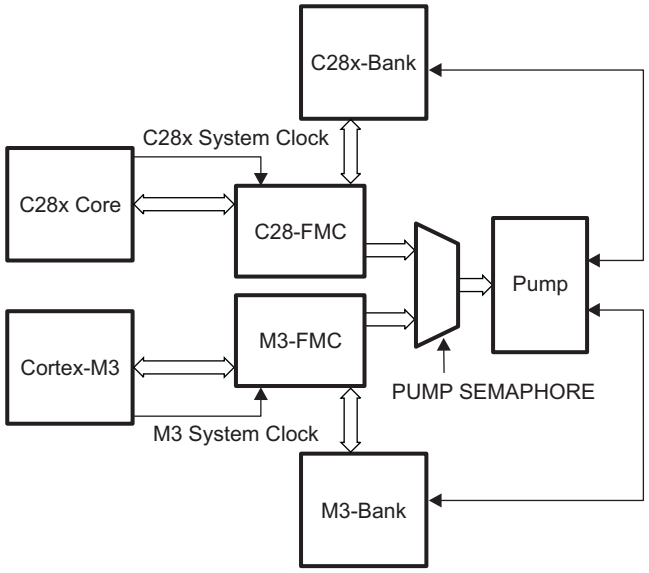
Table 5-83. Programmable OTP Locations in M3 OTP (continued)

Name	Address	Size (x8)	Description
EMACID0	0x680810	4	Default EMAC ID = 0x1A:B6:00:64:00. By programming these two OTP addresses, the user can change default EMAC ID. If left unprogrammed, default EMAC ID = 0x1A:B6:00:64:00
EMACID1	0x680814	4	
CUSTOMER_OPT_MAIN_OSC_CLK_FREQ	0x68081C	4	Input main osc frequency connected to the device. Mboot ROM reads this location to know the MAINOSC frequency to configure the bit rate for CAN / I2C0 and SSI0 master mode / EMAC bootloader. If left unprogrammed,the default input main osc frequency = 20 MHz
OTP BOOTMODE GPIO CONFIG REGISTER	0x680824	4	Used to configure alternate boot mode pins

5.3.5 Flash Module Controller (FMC)

There is a dedicated flash module controller in both the master subsystem (M3-FMC) and the control subsystem (C28x-FMC). The Cortex M3 core in the master subsystem interfaces with the M3 flash module controller (M3-FMC), which in turn, interfaces with the M3 flash bank and shared pump to perform erase/program operations as well as to read data/execute code from the M3 flash bank.

Figure 5-78. FMC Interface with Core, Bank and Pump



The C28x core in the control subsystem interfaces with the C28x flash module controller (C28x-FMC) which in turn, interfaces with the C28x flash bank and shared pump to perform erase/program operations as well as to read data/execute code from the C28x flash bank. Control signals to the flash pump will be controlled by either C28x-FMC or M3-FMC, depending on who gains the flash pump semaphore.

There is a state machine in both M3-FMC and C28x-FMC which generates the erase/program sequences in hardware. This simplifies the Flash API software (refer to the *C2000 F021 Application Programming Interface (API) Reference Guide*, SPNU595, for details on Flash API) which configures control registers in FMC to perform flash erase and program operations.

The following sections (Section 5.3.6, Section 5.3.8, Section 5.3.8, Section 5.3.9, and Section 5.3.10) will describe FMC in detail.

Copyright © 2012–2019, Texas Instruments Incorporated

5.3.6 Flash and OTP Automatic Power-Down Modes

The flash bank and pump consume a significant amount of power when active. The flash module provides a mechanism to automatically power-down flash banks after they have not been accessed for some user-programmable time. Special timers automatically sequence the power-up and power-down of the M3 flash bank and C28x flash bank independently of each other. The shared charge pump module has its own independent power up/down timers as well.

5.3.6.1 Flash/OTP and Pump Power Modes and Wakeup

The flash bank and OTP (M3/C28x) operate in three power modes: Sleep (lowest power), Standby, and Active (highest power)

- Sleep State
  This is the state after a device reset. In this state, the bank is in a sleep state (lowest power). When the flash bank is in the sleep state, a CPU data read or opcode fetch to the flash or OTP memory map area will automatically initiate a change in power modes to the standby state and then to the active state. During this transition time to the active state, the CPU will automatically be stalled.
- Standby State
  This state uses more power than the sleep state, but takes a shorter time to transition to the active or read state. In this state, a CPU data read or opcode fetch to the flash or OTP memory map area will automatically initiate a change in power modes to the active state. During this transition time to the active state, the CPU will automatically be stalled. Once the flash/OTP has reached the active state, the CPU access will complete as normal.
- Active or Read State
  In this state, the bank and pump are in active power mode state (highest power)

The charge pump operates in two power modes:

- Sleep (lowest power)
- Active (highest power)

Any access to a flash bank/OTP causes the charge pump to go into active mode if it is not in sleep mode. Also, any erase or program command causes the charge pump and bank to become active. Also, if any bank is active or in standby mode, the charge pump is active, independent of the charge pump fallback power mode. While pump is in sleep state, a charge pump sleep down counter holds a user configurable value (PSLEEP bit field in FPAC1 register) and when the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP prescaled SYSCLK clock cycles before putting the charge pump into active power mode. Refer to the Section 5.4 for more details.

5.3.6.2 Active Grace Period

The active grace period (AGP) can be used to optimize the flash module power consumption versus access time. Faster access times are associated with higher-power modes of operation. At one extreme, the power control logic could attempt to reduce power consumption by putting the banks and charge pump into a low-power mode immediately at the end of every flash access. However, if accesses are only a few cycles apart, this can actually increase power consumption versus leaving the flash powered, because the banks and charge pump consume more power during flash startup and access.

The active grace periods (supported for M3 flash bank and C28x flash bank independently in addition to the charge pump module) allow the banks and/or charge pump to be maintained in active mode for a specified period following an access. This is done in anticipation of another read within the AGP time, to allow the subsequent read to have a faster access and spend less time dissipating power, than if the bank went into one of the low power modes immediately. If the next access does not occur within the AGP time, the power control logic can automatically put the bank and/or charge pump into a low-power mode to reduce power consumption during long periods of inactivity.

Copyright © 2012–2019, Texas Instruments Incorporated

The AGP value is programmed by a set of programmable counters (FBAC and FPAC2) which keep the flash bank or charge pump in active mode until the counter expires, at which time the bank or charge pump reverts to its fallback power mode as defined in the FBFALLBACK and FPAC1 registers. The application software can program the fallback power mode to be standby or sleep mode to reduce power consumption, or program it to be active mode to keep the bank active regardless of counter settings (default). The charge pump AGP counter remains in its initialized state when any one of the banks is active, including the AGP counter of the bank. The charge pump AGP counter begins counting when both banks have become inactive.

The charge pump can be put into sleep mode only when both the M3-bank and the C28x bank are put to sleep first. The PMPPWR bit in the FPAC1 register of both M3-FMC and C28x-FMC have to be configured for sleep mode for the pump to go to sleep mode. Note that each core has to gain the pump semaphore to be able to write to their respective FMC's PMPPWR bit. As the pump is shared between M3-FMC and C28x-FMC, the effective PAGP value used when powering down the pump will be of the FMC (out of M3-FMC and C28x-FMC) which owns the pump. The pump will become active if an access is made either to the M3-bank or the C28x-bank. Note that the PUMPRDY bit in the M3-FMC register space can be used by the application software to identify the current power mode status of the pump and that PUMPRDY bit (and also PMPPWR bit) in C28x-FMC cannot be used for that purpose. See register descriptions for detailed information.

5.3.7 Flash and OTP Performance

Once flash bank and pump are in the active power state, then a read or fetch access to the bank memory map area can be classified as a flash access (access to an address location in flash) or an OTP access (access to an address location in OTP). Once the CPU throws an access to a flash memory address, data is returned after RWAIT+1 number of core-specific SYSCLK cycles. For an OTP access, data is returned after RWAIT+2 number of core-specific SYSCLK cycles.

RWAIT defines the number of random access wait states and is configurable using the RWAIT bit-field in the FRDCNTL register. At reset, the RWAIT bit-field value defaults to a worst-case wait state count (15) and thus, needs to be initialized for the appropriate number of wait states to improve performance, based on the CPU clock rate and the access time of the flash.

The flash supports 0-wait accesses when the RWAIT bits are set to zero. This assumes that the CPU speed is low enough to accommodate the access time.

For a given system clock frequency, RWAIT must be configured using the below formula:

RWAIT =[(SYSCLK/FCLK)-1] round up to the next highest integer or 1, whichever is larger
where
SYSCLK is the system operating frequency

FCLK is flash clock frequency. FCLK should be ≤ FCLKmax, allowed maximum flash clock frequency with RWAIT=0.

If RWAIT results in a fractional value when calculated using the above formula, RWAIT has to be rounded up to the nearest integer.

5.3.8 Flash Read Interface

This section provides details about the data read modes to access flash bank/OTP and the configuration registers which control the read interface. In addition to a standard read mode, FMCs have built-in prefetch and cache mechanisms to allow increased clock speeds and CPU throughput wherever applicable.

5.3.8.1 M3-FMC Flash Read Interface

5.3.8.1.1 Standard Read Mode

Standard read mode is defined as the read mode in effect when program cache/prefetch-mechanism and data cache are disabled. It is also the default read mode after reset. During this mode, each read access to flash is decoded by the flash wrapper to fetch the data from the addressed location and the data is returned after the RWAIT+1 number of cycles.

Copyright © 2012–2019, Texas Instruments Incorporated

The program cache/prefetch mecahnism and data cache are bypassed in standard read mode; therefore, every access to the flash/OTP is used by the CPU immediately and every access creates a unique flash bank access.

Standard read mode is the recommended mode for lower system frequency operation in which RWAIT can be set to zero to provide single cycle access operation. FMC can operate at higher frequencies using standard read mode at the expense of adding wait states. At higher system frequencies, it is recommended to enable cache and prefetch mechanisms to improve performance. Please refer to device specific data manual to determine maximum flash frequency allowed in standard read mode (i.e., maximum flash clock frequency with one wait state - FCLKmax).

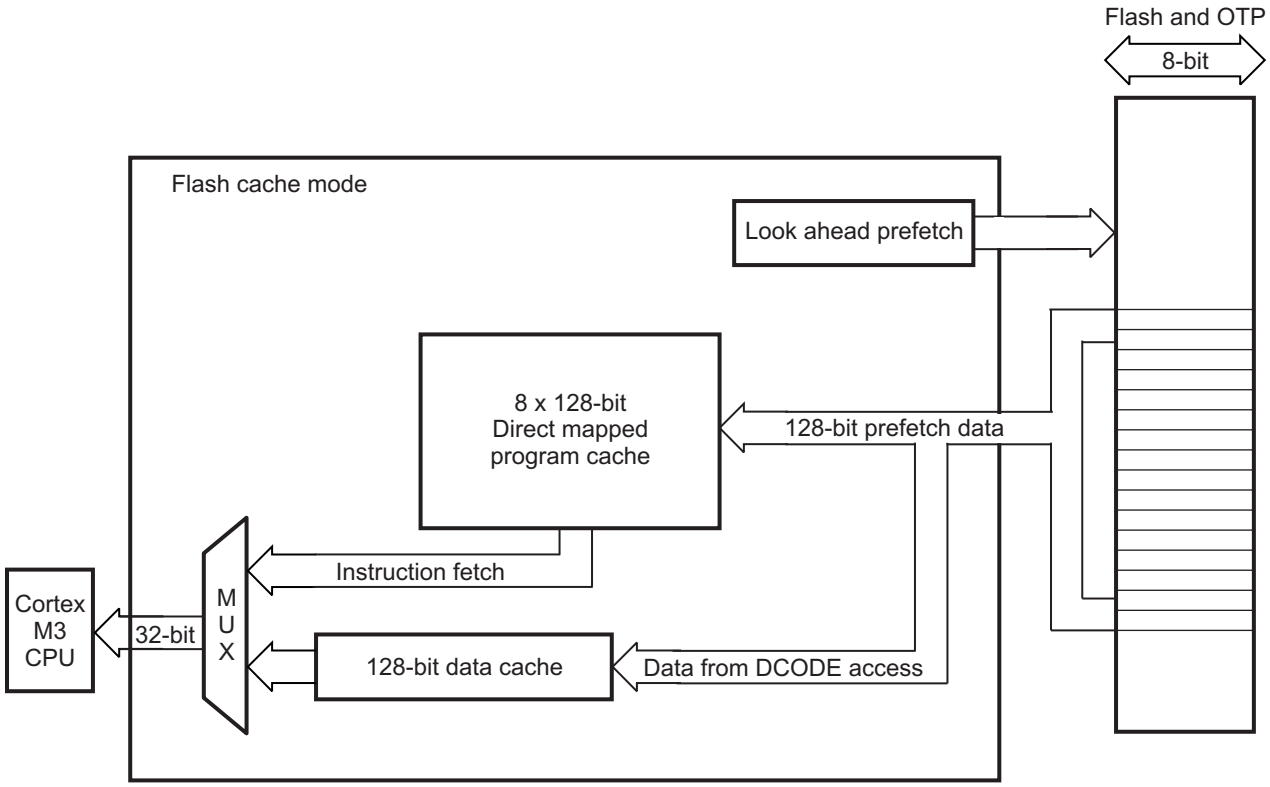
5.3.8.1.2 Cache Mode

5.3.8.1.2.1 Program Cache

Flash memory is typically used to store application code. During code execution, instructions are fetched from sequential memory addresses, except when a discontinuity occurs. Usually, the portion of the code that resides in sequential addresses makes up the majority of the application code, and is referred to as linear code. To improve the performance of linear code execution, a flash prefetch-mechanism has been implemented in FMC. This prefetch mechanism does a look-ahead prefetch on linear address increments starting from the address of the last program access.

Apart from linear code, in general application codes, there may be several loops wherein a set of instructions located in sequential addresses are executed repeatedly in a loop, until a condition holds true. To improve the performance of small loop code execution, an 8-level deep 128-bit wide (8 x 128) direct mapped program cache has been implemented in the FMC. Whenever instructions in cache are fetched for CPU processing, the flash prefetch mechanism does a look-ahead prefetch of 128 bits from the next linear 128-bit aligned address from last address access, and fills the program cache as shown in Figure 5-79.

Figure 5-79. Flash Cache Mode



Copyright © 2012–2019, Texas Instruments Incorporated

Up to four 32-bit instructions or up to eight 16-bit instructions can reside within a single 128-bit access. For every 128-bit instruction fetch from the flash bank, it is likely that there are up to eight instructions (assuming 16-bit instructions) in each level of cache, ready to process through the CPU. During the time it takes to process these instructions, the flash prefetch mechanism automatically initiates another access to the flash bank to prefetch the next 128 bits. In this manner, the flash prefetch mechanism works in the background to keep the cache as full as possible with data corresponding to the next linear address. Using this technique, the overall efficiency of sequential code execution and small loop code execution from flash or OTP is improved significantly.

The flash program-cache and prefetch mechanism features are disabled by default. Setting the PROG_CACHE_EN bit in the FRD_INTF_CTRL register enables this cache mode. This flash prefetch and cache mechanisms are independent of the CPU pipeline.

When Cortex M3 in the master subsystem initiates an ICODE access from an address in program space in flash:

1. M3-FMC checks the program cache, if enabled, to determine the presence of required data.
2. If data corresponding to the requested address is present in the cache, it is construed as a cache hit and data will be provided to the CPU from cache. At this time, the prefetch mechanism will fetch 128 bits from the next linear 128-bit aligned address from last address access, and fills the program cache.
3. If data corresponding to the requested address is not present in the cache, it is construed as a cache miss. Therefore, the prefetch mechanism will fetch 128 bits of the data from the requested address in bank (the starting address of the access from flash is automatically aligned to a 128-bit boundary such that the instruction location is within the 128 bits to be fetched) and writes that data on to cache, and eventually the requested data is sent to the CPU from cache for processing.

If cache mode is enabled, then the last row of 128 bits in bank should not be used, because the prefetch logic which does a look-ahead prefetch, will try to fetch from outside the bank/OTP and would result in an ECC error.

The flash prefetch mechanism is aborted only on a PC discontinuity caused by executing an instruction such as a branch, function call, or loop, and so on. When this occurs, the prefetch is aborted. There are two possible scenarios when this occurs:

1. If the destination address is within the flash or OTP, then prefetch aborts and then resumes at the destination address.
2. If the destination address is outside of the flash and OTP, the prefetch is aborted and begins again only when a branch is made back into the flash or OTP. The flash prefetch mechanism only applies to instruction fetches (ICODE) from program space. Data space accesses (DCODE) do not utilize the prefetch mechanism. For data space accesses, the program cache and prefetch mechanism are bypassed. If an instruction prefetch is already in progress when a data read operation is initiated, then the data read will be stalled until the prefetch completes.

5.3.8.1.2.2 Data Cache

Along with the program cache, a data cache of 128 bits width is also implemented to improve data space access (DCODE) performance. This data cache will not be filled by the prefetch mechanism. When any kind of data space access is made by the CPU from an address in the bank, and if the data corresponding to the requested address is not in the data cache, then 128 bits of data will be read from the bank and loaded in the data cache. The data is eventually sent to the CPU for processing. The starting address of the access from flash is automatically aligned to a 128-bit boundary such that the requested address location is within the 128 bits to be read from the bank. By default, this data cache is disabled and can be enabled by setting DATA_CACHE_EN bit in the FRD_INTF_CTRL register.

Some other points to keep in mind when working with M3 flash:

- Reads of the CSM password locations, GRABRAM, GRABSECT, ECSLKEY and EXEONLY locations are hardwired for 10 wait states. The RWAIT bits have no effect on these locations.
- CPU writes to the flash or OTP memory map areas are ignored. They complete in a single cycle.
- DCODE access to zone-1 and zone-2 password locations would return the data to Code Security Module (CSM) and return a value of zero to Cortex-M3 when the respective password lock bits are not all 1s and the respective zone is not unlocked.
- ICODE accesses to zone-1 and zone-2 password locations return a 0 when the respective password

Copyright © 2012–2019, Texas Instruments Incorporated

lock bits are not all 1s and the respective zone is not unlocked.

- When the Code Security Module (CSM) is secured, reads to the flash/OTP memory map area from outside the secure zone take the same number of cycles as a normal access. However, the read operation returns a zero.
- The arbitration scheme in M3-FMC prioritizes Cortex-M3 accesses in the fixed priority order of data space access (DCODE), program space access (ICODE)/program space prefetch.
- When FSM interface is active for erase/program operations, data in the program cache and data cache in FMC will be invalidated.

5.3.8.2 C28x-FMC Flash Read Interface

5.3.8.2.1 Standard Read Mode

Standard read mode is defined as the read mode in effect when code prefetch-mechanism and data cache are disabled. It is also the default read mode after reset. During this mode, each read access to flash is decoded by the flash wrapper to fetch the data from the addressed location and the data is returned after the RWAIT+1 number of cycles.

Prefetch buffers associated with the prefetch mechanism and data cache are bypassed in standard read mode; therefore, every access to the flash/OTP is used by the CPU immediately and every access creates a unique flash bank access.

Standard read mode is the recommended mode for lower system frequency operation in which RWAIT can be set to zero to provide single cycle access operation. The FMC can operate at higher frequencies using standard read mode at the expense of adding wait states. At higher system frequencies, it is recommended to enable the code prefetch mechanism and data cache to improve performance. Please refer to the device specific data manual to determine maximum flash frequency allowed in standard read mode (that is, maximum flash clock frequency with one wait state - FCLKmax).

5.3.8.2.2 Prefetch Mode

Flash memory is typically used to store application code. During code execution, instructions are fetched from sequential memory addresses, except when a discontinuity occurs. Usually the portion of the code that resides in sequential addresses makes up the majority of the application code and is referred to as linear code. To improve the performance of linear code execution, a flash prefetch-mechanism has been implemented in FMC. Figure 5-80 illustrates how this mode functions.

This prefetch mechanism does a look-ahead prefetch on linear address increments starting from the address of the last instruction fetch. The flash prefetch mechanism is disabled by default. Setting the PREFETCH_EN bit in the FRD_INTF_CTRL register enables this prefetch mode.

An instruction fetch from the flash or OTP reads out 128 bits per access. The starting address of the access from flash is automatically aligned to a 128-bit boundary, such that the instruction location is within the 128 bits to be fetched. With flash prefetch mode enabled, the 128 bits read from the instruction fetch are stored in a 128-bit wide by 2-level deep instruction prefetch buffer. The contents of this prefetch buffer are then sent to the CPU for processing as required.

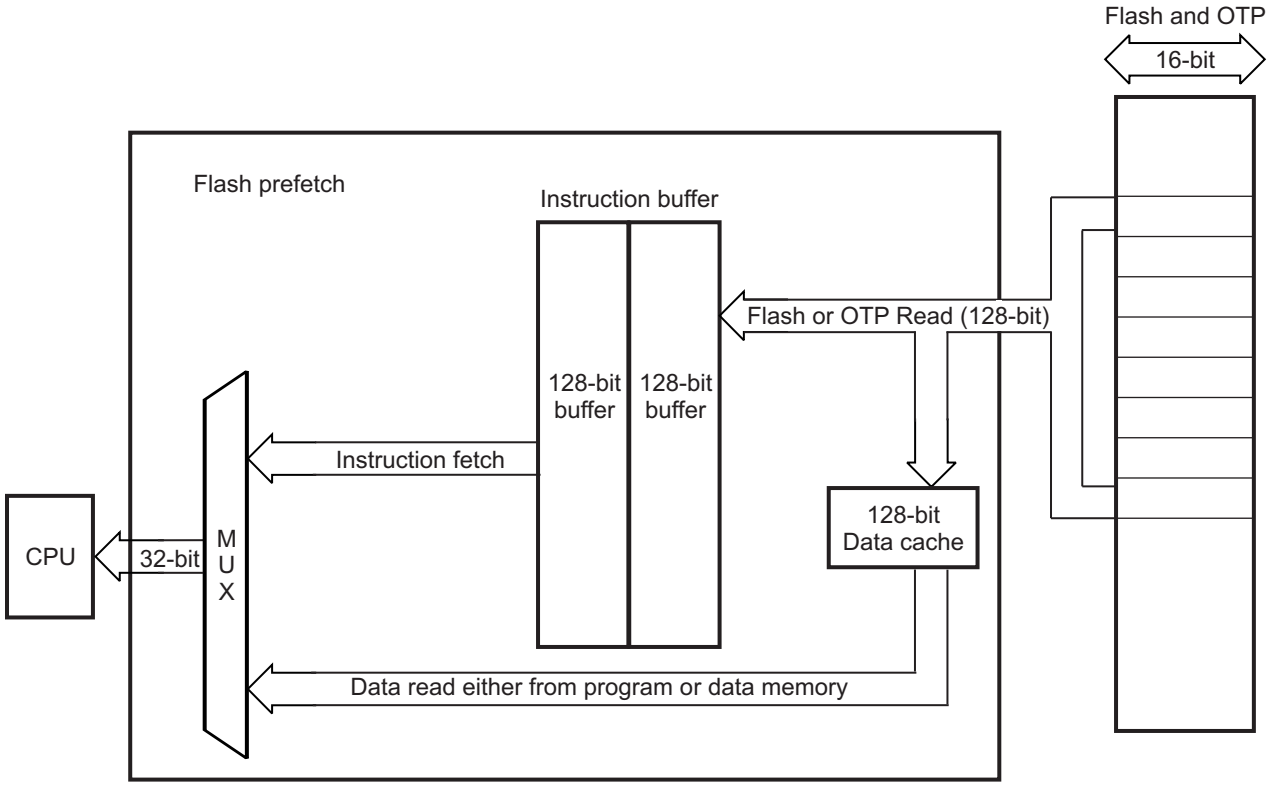
Up to four 32-bit instructions or up to eight 16-bit instructions can reside within a single 128-bit access. The majority of C28x instructions are 16 bits, so for every 128-bit instruction fetch from the flash bank, it is likely that there are up to eight instructions in the prefetch buffer ready to process through the CPU. During the time it takes to process these instructions, the flash prefetch mechanism automatically initiates another access to the flash bank to prefetch the next 128 bits. In this manner, the flash prefetch mechanism works in the background to keep the instruction prefetch buffers as full as possible. Using this technique, the overall efficiency of sequential code execution from flash or OTP is improved significantly. If the prefetch mechanism is enabled, then the last row of 128-bits in the bank should not be used, because the prefetch logic which does a look-ahead prefetch, will try to fetch from outside the bank and would result in ECC error.

The flash prefetch is aborted only on a PC discontinuity caused by executing an instruction such as a branch, BANZ, call, or loop. When this occurs, the prefetch is aborted and the contents of the prefetch buffer are flushed. There are two possible scenarios when this occurs:

Copyright © 2012–2019, Texas Instruments Incorporated

1. If the destination address is within the flash or OTP, the prefetch aborts and then resumes at the destination address.
2. If the destination address is outside of the flash and OTP, the prefetch is aborted and begins again only when a branch is made back into the flash or OTP. The flash prefetch mechanism only applies to instruction fetches from program space. ata reads from data memory and from program memory do not utilize the prefetch buffer capability and thus bypass the prefetch buffer. For example, instructions such as MAC, DMAC, and PREAD read a data value from program memory. When this read happens, the prefetch buffer is bypassed but the buffer is not flushed. If an instruction prefetch is already in progress when a data read operation is initiated, then the data read will be stalled until the prefetch completes.

Figure 5-80. Flash Prefetch Mode



5.3.8.2.2.1 Data Cache

Along with the prefetch mechanism, a data cache of 128 bits width is also implemented to improve data space read and program space read performance. This data cache will not be filled by the prefetch mechanism. When any kind of data-space read or program-space read is made by the CPU from an address in the bank, and if the data corresponding to the requested address is not in the data cache, then 128 bits of data will be read from the bank and is loaded in the data cache. This data is eventually sent to the CPU for processing. The starting address of the access from flash is automatically aligned to a 128-bit boundary such that the requested address location is within the 128 bits to be read from the bank. By default, this data cache is disabled and can be enabled by setting DATA_CACHE_EN bit in the FRD_INTF_CTRL register.

Some other points to keep in mind when working with C28x flash:

- Reads of the CSM password locations, ECSLKEY and EXEONLY locations are hardwired for 10 wait-states. The RWAIT bits have no effect on these locations
- CPU writes to the flash or OTP memory map areas are ignored. They complete in a single cycle.
- If C28x security zone is in the locked state and the respective password lock bits are not all 1s, then,
  - Data reads to C28X-Z1-CSMPSWD, C28x-Z1-ECSLPSWD will return 0.

Copyright © 2012–2019, Texas Instruments Incorporated

  - Program space reads to C28X-Z1-CSMPSWD, C28x-Z1-ECSLPSWD will return 0.
  - Program fetches to C28X-Z1-CSMPSWD, C28x-Z1-ECSLPSWD will return 0.
- When the code security module (CSM) is secured, reads to the flash/OTP memory map area from outside the secure zone take the same number of cycles as a normal access. However, the read operation returns a zero.
- The arbitration scheme in C28x-FMC prioritizes C28x core accesses in the fixed priority order of data read (highest priority), program space read and program fetches/program prefetches (lowest priority)
- When FSM interface is active for erase/program operations, data in the prefetch buffers and data cache in FMC will be flushed.

5.3.9 Erase/Program Flash

The F021 Flash (F28M35x M3 Flash and C28x Flash) should be programmed, erased, and verified only by using the F021 Flash API library. These functions are written, compiled and validated by Texas Instruments. The flash module contains a flash state machine (FSM) to perform, program, and erase operations. This section only provides a high-level description for these operations. Please refer to the *C2000 F021 Flash Application Programming Interface (API) Reference Guide*, SPNU595, for more information. Please note that Flash API functions must only be executed from RAM and not from flash.

A typical flow to program flash is:
Erase → Program → Verify

5.3.9.1 Erase

When the target flash is erased, it reads as all 1's. This state is called 'blank.' The erase function must be executed before programming. The user should NOT skip erase on sectors that read as 'blank' because these sectors may require additional erasing due to marginally erased bits columns. The FSM provides an “Erase Sector” command to erase the target sector. The erase function erases the data and the ECC together. These commands are implemented by the following Flash API function:

Fapi_issueAsyncCommandWithAddress();

The Flash API provides the following function to determine if the flash bank is 'blank':

Fapi_doBlankCheck();

5.3.9.2 Program

The FSM provides command to program the customer OTP area and flash program data area. This command is also used to program ECC check bits.

This command is implemented by the following Flash API functions:

Fapi_issueProgrammingCommand();

The Program function provides the options to program data without ECC, data along with user provided ECC data, data along with ECC calculated by API software, and to program ECC only.

5.3.9.3 Verify

After programming, the user must perform verify using API function Fapi_doVerify(). This function verifies the flash contents against supplied data in normal read mode and also in read-margin modes. Read margin modes 0 and 1 are used to test cells for marginality. Read margin 0 is used to test cells for marginality of programmed cells. Read margin 1 is used to test cells for marginality of erased cells.

Application software is generally developed to enter read-margin mode during power-up and to perform a full CRC check of the flash contents. In this way, a potential read mode failure can be identified prior to data loss or gain causing a bit flip.

NOTE: The read-margin modes are intended for diagnostic capabilities. Flash content is guaranteed for the lifetime specified in the datasheet.

Copyright © 2012–2019, Texas Instruments Incorporated

Read margin modes should only be entered when executing from RAM. Banks must be powered up before entering a read margin mode. When entering a read margin mode, or changing from one read margin mode to the other, allow 1us delay before the first flash access to allow the flash banks to adjust to the new mode.

Read margin modes 0/1 are enabled by setting the RM0/RM1 bits in the special read control register, FSPRD. The recommended procedures to enter, change, and exit read margin mode are shown below.

To enter read-margin mode:

1. Start execution out of RAM.
2. Set the banks to remain powered up by writing 3 to bits 0:1 of the FBFALLBACK register.
3. Turn on read margin 0 or 1 by writing 1 or 2 to the FSPRD register.
4. Do one dummy read from each bank to turn on the bank.
5. Flush the data cache by reading two flash locations separated by at least 32 bytes. The read data should be ignored for these reads.
6. Wait 1 us

Any reads will now be with the selected read margins. The application can now return to flash if desired.

To exit read-margin mode:

1. Start execution out of RAM
2. Turn off read margin 0 or 1 by writing 0 to the FSPRD register
3. Flush the data cache by reading two flash locations separated by at least 32 bytes. The read data should be ignored for these reads.
4. Wait 1 µs

Any reads will now be a standard read. The application can now return to flash if desired. The application can also set the FBFALLBACK register to the original values.

To change one read margin mode to the other:

1. Follow the procedure above to exit read margin mode.
2. Follow the procedure above to enter read margin mode.

5.3.10 Error Correction Code (ECC) Protection

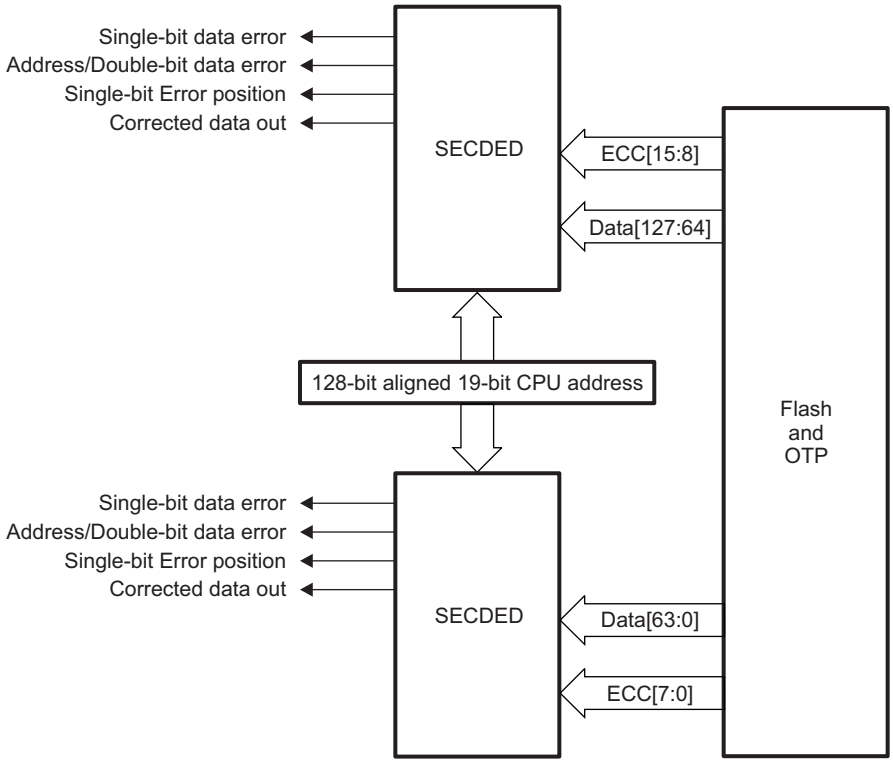
M3-FMC and C28x-FMC contain an embedded single error correction and double error detection (SECDED) module. SECDED, when enabled, provides the capability to screen out memory faults. SECDED can detect and correct single-bit data errors and detect address errors/double-bit data errors. For every 64 bits of flash/OTP data (aligned on a 64-bit memory boundary) that is programmed, eight ECC check bits have to be calculated and programmed in ECC memory space. Refer to the device data manual for the flash/OTP ECC memory map.

SECDED works with a total of eight user-calculated error correction code (ECC) check bits associated with each 64-bit wide data word and its corresponding 128-bit memory-aligned address. Users must program ECC check bits along with flash data. TI recommends using the AutoEccGeneration option available in the Plugin/API to program the ECC. Figure 5-81 illustrates the ECC logic inputs and outputs.

The Flash API uses hardware ECC logic in the device to generate the ECC data for the given flash data. The Flash plugin and UniFlash use Flash APIs to generate and program ECC data.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 5-81. ECC Logic Inputs and Outputs



During an instruction fetch or data read operation, the 19 address bits (three least significant bits of address are not considered), together with the 64-bit data/8-bit ECC read-out of flash banks/ECC memory map area, pass through the SECDED logic and the eight check bits are produced in FMC. These eight calculated ECC check bits are then XORed with the stored check bits (user-programmed check bits) associated with the address and the read data. The 8-bit output is decoded inside the SECDED module to determine one of three conditions:

- No error occurred
- A correctable error (single bit data error) occurred
- A non-correctable error (double bit data error or address error) occurred

If the SECDED logic finds a single bit error in the address field, then it is considered to be a non-correctable error.

NOTE: Since ECC is calculated for an entire 64-bit data, a non 64-bit read such as a byte read or a half-word read will still force the entire 64-bit data to be read and calculated, but only the byte or half-word will be actually used by the CPU.

This ECC (SECDED) feature is enabled at reset. The ECC_ENABLE register can be used to configure (enable or disable) the ECC feature. The ECC for the application code must be programmed.

There are two SECDED modules in each FMC. Out of the 128-bit data (aligned on a 128-bit memory boundary) read from the bank/OTP address, the lower 64-bits of data and corresponding 8 ECC bits (read from user programmable ECC memory area) are fed as inputs to one SECDED module along with the 128-bit aligned 19-bit address from where data has been read. The upper 64-bits of data and corresponding 8 ECC bits are fed as inputs to another SECDED module in parallel along with 128-bit aligned 19-bit address. Each of the SECDED modules evaluate their inputs and determine if there is any single bit data error or double bit data error/address error.

ECC logic will be bypassed when the 64 data bits and the associated ECC bits fetched from the bank are either all ones or zeros.

Copyright © 2012–2019, Texas Instruments Incorporated

ECC logic should be disabled before reading the ECC memory space and it can be enabled as needed after reading the ECC memory space.

5.3.10.1 Single-Bit Data Error

This section provides information for both single-bit, bit data errors or single-bit ECC check bit errors. If there is a single bit flip (0 to 1 or 1 to 0) in flash data or in ECC data, then it is considered as a single-bit data error. The SECDED module detects and corrects single bit errors, if any, in the 64-bit flash data or eight ECC check bits read from the flash/ECC memory map before the read data is provided to the CPU.

When SECDED finds and corrects single bit data errors, the following information is logged in ECC registers if the ECC feature is enabled:

- Address where the error occurred (SINGLE_ERR_ADDR register).
- Whether error occurred in data bits or ECC bits (ERR_TYPE bit field in ERR_POS register)
- Bit position at which error occurred (ERR_POS bit field in ERR_POS register)
- Whether the error occurred in lower 64-bit data/ECC or in upper 64-bit data/ECC (ECC_L_OR_H field in ERR_POS register)
- Whether the corrected value is 0 (FAIL_0 flag in ERR_STATUS register)
- Whether the corrected value is 1 (FAIL_1 flag in ERR_STATUS register)
- A single bit error counter that increments on every single bit error occurrence (ERR_CNT register) until a user configurable threshold (see ERR_THRESHOLD) is met
- A flag that gets set when one more single bit error occurs after ERR_CNT equals ERR_THRESHOLD (SINGLE_ERR_INT_FLG flag in ERR_INTFLG register)

When the ERR_CNT value equals THRESHOLD+1 value and a single bit error occurs, the SINGLE_ERR_INT flag is set, and an interrupt (C28FLSINGERR on C28x PIE and M3 Flash Single Error on M3 NVIC have to be enabled for interrupts, if needed) is fired. The SINGLE_ERR interrupt will not be fired again until the SINGLE_ERR_INT_FLG is cleared. If the single error interrupt flag is not cleared using the corresponding error interrupt clear bit in the ERR_INTCLR register, the error interrupt will not come again, as this is an edge-based interrupt.

5.3.10.2 Uncorrectable Error

Uncorrectable errors include address errors and double-bit errors in data/ECC. When SECDED finds uncorrectable errors, the following information is logged in ECC registers if the ECC feature is enabled:

- Address where the error occurred (UNC_ERR_ADDR register).
- A flag is set indicating that an uncorrectable error occured (UNC_ERR flag in ERR_STATUS register)
- A flag is set indicating that an uncorrectable error interrupt is fired (UNC_ERR_INT_FLG in ERR_INTFLG register)

When an uncorrectable error occurs, the UNC_ERR_INT_FLG bit is set and an uncorrectable error interrupt is fired. This uncorrectable error interrupt generates a bus fault on Cortex-M3 and an NMI on the C28x core, if enabled. If an uncorrectable error interrupt flag is not cleared using the corresponding error interrupt clear bit in the ERR_INTCLR register, an error interrupt will not come again, as this is an edge-based interrupt.

Although the ECC is calculated on a 64-bit basis, a read of any address location within a 128-bit aligned flash memory will cause the uncorrectable error flag to get set when there is an uncorrectable error in both or in either one of the lower 64 and upper 64 bits (or corresponding ECC check bits) of that 128-bit data. NMI will occur on C28x for a read of any address location within a 128-bit aligned flash memory when there is an uncorrectable error in both or in either one of the lower 64 and upper 64 bits (or corresponding ECC check bits) of that 128-bit data. However, the bus-fault on M3 will occur only when a memory location from the particular 6 -bits (or corresponding ECC check bits) that have an uncorrectable error is read. They will not occur for the other 64 bits that do not have an uncorrectable error in the 128-bit memory aligned data.

Copyright © 2012–2019, Texas Instruments Incorporated

5.3.10.3 SECDED Logic Correctness Check

Since error detection and correction logic is part of safety-critical logic, safety applications may need to ensure that the SECDED logic is always working properly. For these safety concerns, in order to ensure the correctness of the SECDED logic, an ECC test mode is provided to test the correctness of ECC logic periodically. In this ECC test mode, instead of a data/ECC read from bank and address given to the bank for read, a user-configurable 64-bit data/8-bit ECC check bits (FDATAH_TEST, FDATAL_TEST, FECC_TEST) and address (FADDR_TEST) are fed to SECDED logic. Using this test mode, users can introduce single-bit errors or double-bit errors or address errors and check whether SECDED logic is catching those errors or not. Users can also check if SECDED logic is reporting any false errors when no errors are introduced.

This ECC test mode can be enabled by setting the ECC_TEST_EN bit in the FECC_CTRL register. When the ECC test mode is enabled, the CPU cannot read the data from flash and instead the CPU gets data from the ECC test mode registers (FDATAH_TEST/FDATAL_TEST). This is because ECC test mode registers (FDATAH_TEST, FDATAL_TEST, FECC_TEST) are multiplexed with data from the flash. Hence, the CPU should not read/fetch from flash when ECC test mode is enabled. For this reason, ECC test mode code should be executed from RAM and not from flash.

Only one of the SECDED modules (out of the two SECDED modules that work on the lower 64 bits and upper 64 bits of a read 128-bit data) at a time can be tested. The ECC_SELECT bit in the FECC_CTRL register can be configured by users to select one of the SECDED modules for test.

To test the ECC logic using ECC test mode, follow the steps below:

1. Obtain the ECC for a given flash address (64-bit aligned) and 64-bit data by using Fapi_calculateECC() function provided in Flash API. For the C28x subsystem, remember that this function needs a byte address and hence left shift the address by one position before passing the address to this function. For the M3 subsystem, the address can be passed as is.
2. Develop an application to test ECC logic using the above data. In this application:
   - For the single-bit error test, set the single-bit error threshold to 0. This step is not required for uncorrectable errors.
   - Write the 64-bit aligned, 19-bit flash address in the FADDR_TEST register.
   - Write 64-bit data in the FDATAx_TEST register.
   - Write the corresponding 8-bit ECC in the FECC_TEST register.
   - In any of the above three steps, users can insert errors (single-bit data error or double-bit data error or address error or single-bit ECC error or double-bit ECC error) so that they can check whether or not ECC logic is able to catch the errors.
   - Select the ECC logic block (lower 64 bits or upper 64 bits) which needs to be tested using the ECC_SELECT bit in the FECC_CTRL register.
   - Enable ECC test mode using the ECC_TEST_EN bit in the FECC_CTRL register to enable ECC test logic to evaluate the address, data, ECC in FADDR_TEST, FDATAx_TEST and FECC_TEST registers for ECC errors.
   - In the flash single-bit error ISR and in the NMI ISR, disable the ECC test mode (ECC_TEST_EN = 0). This is important because the ECC test mode, when enabled, will continuously evaluate the test mode registers for errors in every cycle and hence, single-bit error interrupt and/or NMI will occur continuously until the ECC test mode is disabled.

Once the above ECC test mode registers are written by the user:

- The FECC_OUTH register holds the data output bits 63:32 from the SECDED block under test
- The FECC_OUTL register holds the data output bits 31:0 from the SECDED block under test
- The FECC_STATUS register holds the status of single-bit error occurrence, uncorrectable error occurrence, and error position of single bit error in data/check bits

5.3.11 Reserved Locations Within Flash and OTP

When allocating code and data to flash and OTP memory, keep the following reserved locations of flash/OTP in mind:

- M3 OTP has reserved locations for OTPSECLOCK, Zone 2 Flash entry point and EMAC ID whose

Copyright © 2012–2019, Texas Instruments Incorporated

address locations and sizes are mentioned earlier. For more details on the functionality of these fields, please refer to the *System Control and Interrupts* chapter and the *ROM Code and Peripheral Booting* chapter.

- Refer to the Code Security Module (CSM) section in the *System control and Interrupts* chapter for reserved locations in M3 flash for Zone 1 and Zone 2 CSM password, Zone 1 and Zone 2 ECSL passwords, Zone 1 and Zone 2 GRABSECT, GRABRAM, and EXEONLY fields.
- Refer to the CSM section in the *System Control and Interrupts* chapter for reserved locations in C28x flash for CSM password, ECSL password, and EXEONLY field.
- Refer to the *ROM Code and Peripheral Booting* chapter for reserved locations in M3 flash and C28x flash for boot to flash entry point. These locations are reserved for an entry into flash branch instruction. When the boot to flash boot option is used, the boot ROM will jump to these addresses in flash. If the user programs a branch instruction here, that will then re-direct code execution to the entry point of the application.
- For code security operation, all address ranges (inclusive) given below in M3 Flash cannot be used for program code or data, but must be programmed to 0x00 when the Code Security Password is programmed in M3 Flash. If security is not a concern, then these addresses may be used for code or data. See the *System Control and Interrupts* chapter for information in using the CSM.
  - From 0x00200024 to 0x0020002F
  - From 0x00200050 to 0x002000 FF
  - From 0x0027FF00 to 0x0027FFDB
- For code security operation, all addresses from 0x0013FF80 to 0x0013FFEF (inclusive of these addresses) in C28x Flash cannot be used for program code or data, but must be programmed to 0x0000 when the Code Security Password is programmed in C28x Flash. If security is not a concern, then these addresses may be used for code or data. See the *System Control and Interrupts* chapter for information on using the CSM.

5.3.12 Procedure to Change the Flash Control Registers

During flash configuration, no accesses to the flash or OTP can be in progress. This includes instructions still in the CPU pipeline, data reads, and instruction prefetch operations. To be sure that no access takes place during the configuration change, you should follow the procedure shown below for any code that modifies the flash control registers. This is applicable for both master subsystem and control subsystem application.

1. Start executing application code from RAM/Flash/OTP.
2. Branch to or call the flash configuration code (that writes to flash control registers) in RAM. This is required to properly flush the CPU pipeline before the configuration change. (The function that changes the flash configuration cannot execute from the flash or OTP. It must reside in RAM .
3. Execute the Flash configuration code (should be located in RAM) that writes to flash control registers like FRDCNTL, FRD_INTF_CTRL etc.
4. At the end of the flash configuration code execution, wait eight cycles to let the write instructions propagate through the CPU pipeline. This must be done before the return-from-function call is made.
5. Return to the calling function which might reside in RAM or Flash/OTP and continue execution.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4 Flash Registers

The M3 flash/OTP memory and C28x flash can be configured by the registers shown in Table 5-84 and Table 5-85, respectively. All the control subsystem flash registers are protected by the code security module. For master subsystem flash registers, the SECZONEREQUEST semaphore register (mapped at address 0x400FA160) is provided for accessing the flash registers for the two M3 security zones, with out compromising on security.

The flash control registers should not be written to by code that is running from OTP or flash memory or while an access to flash or OTP is in progress. All register accesses to the flash registers should be made from code executing outside of flash/OTP memory; an access should not be attempted until all activity on the flash/OTP has completed. No hardware is included to protect against this. To summarize, you can read the flash registers from code executing in flash/OTP; however, do not write to the registers.

CPU write access to the flash registers can be enabled only by executing the EALLOW instruction on the C28x core and by initializing MWRALLOW with 0xA5A5A5A5 on Cortex-M3. Write access to flash registers is disabled when the EDIS instruction is executed on the C28x core. For Cortex-M3, write access to flash registers is disabled when MWRALLOW is initialized with any value other than 0xA5A5A5A5. This protects the registers from spurious accesses. The registers can be accessed through the JTAG port without the need to execute EALLOW/initialize MWRALLOW.

Table 5-84. Flash Registers Memory Map on Master Subsystem

Register Acronym	Register Description	Size (x8)	Type	M3 Offset (0x8)	M3 Protection	Reset Source
Flash Control Registers				**0x400F:A000**		
FRDCNTL	Flash Read Control Register	4	R/W	0x0	MWRALLOW	M3SYSRSTn
FSPRD	Flash read margin control register	4	R/W	0x4	MWRALLOW	M3SYSRSTn
Reserved	Reserved	52				
FBAC	Flash Bank Access Control Register	4	R/W	0x03C	MWRALLOW	M3SYSRSTn
FBFALLBACK	Flash Bank Fallback Power Register	4	R/W	0x040	MWRALLOW	M3SYSRSTn
FBPRDY	Flash Bank Pump Ready Register	4	R	0x044	MWRALLOW	M3SYSRSTn
FPAC1	Flash Pump Access Control Register 1	4	R/W	0x048	MWRALLOW	M3SYSRSTn
FPAC2	Flash Pump Access Control Register 2	4	R/W	0x04C	MWRALLOW	M3SYSRSTn
FMAC	Flash Module Access Control Register	4	R	0x050	MWRALLOW	M3SYSRSTn
FMSTAT	Flash Module Status Register (Used with Flash API – Refer to Flash Application Programming Interface User's Specification for details of this register)	4	R	0x054	MWRALLOW	M3SYSRSTn
Reserved	Reserved	264				

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-84. Flash Registers Memory Map on Master Subsystem (continued)

Register Acronym	Register Description	Size (x8)	Type	M3 Offset (0x8)	M3 Protection	Reset Source
SECZONEREQUEST	Security Zone Semaphore for M3 Flash Wrapper registers	4	R/W	0x160	MWRALLOW	M3SYSRSTn
Reserved	Reserved	412				
FRD_INTF_CTRL	Flash Read Interface Control Register	4	R/W	0x300	MWRALLOW	M3SYSRSTn
Flash ECC/Error Log Registers				**0x400F:A600**		
ECC_ENABLE	ECC Enable Register	4	R/W	0x0	MWRALLOW	M3SYSRSTn
SINGLE_ERR_ADDR	Single Error Address Register	4	R	0x4	MWRALLOW	M3SYSRSTn
UNC_ERR_ADDR	Uncorrectable Error Address Register	4	R	0x8	MWRALLOW	M3SYSRSTn
ERR_STATUS	Error Status Register	4	R	0xC	MWRALLOW	M3SYSRSTn
ERR_POS	Error Position Register	4	R	0x10	MWRALLOW	M3SYSRSTn
ERR_STATUS_CLR	Error Status Clear Register	4	R/W0-1	0x14	MWRALLOW	M3SYSRSTn
ERR_CNT	Error Counter Register	4	R	0x18	MWRALLOW	M3SYSRSTn
ERR_THRESHOLD	Error Threshold Register	4	R/W	0x1C	MWRALLOW	M3SYSRSTn
ERR_INTFLG	Error Interrupt Flag Register	4	R	0x20	MWRALLOW	M3SYSRSTn
ERR_INTCLR	Error Interrupt Flag Clear Register	4	R/W0-1	0x24	MWRALLOW	M3SYSRSTn
FDATAH_TEST	Data High Test Register	4	R/W	0x28	MWRALLOW	M3SYSRSTn
FDATAL_TEST	Data Low Test Register	4	R/W	0x2C	MWRALLOW	M3SYSRSTn
FADDR_TEST	ECC Test Address Register	4	R/W	0x30	MWRALLOW	M3SYSRSTn
FECC_TEST	ECC Test Register	4	R/W	0x34	MWRALLOW	M3SYSRSTn
FECC_CTRL	ECC Control Register	4	R/W	0x38	MWRALLOW	M3SYSRSTn
FECC_FOUTH_TEST	Test Data Out High Register	4	R	0x3C	MWRALLOW	M3SYSRSTn
FECC_FOUTL_TEST	Test Data Out Low Register	4	R	0x40	MWRALLOW	M3SYSRSTn
FECC_STATUS	ECC Status Register	4	R	0x44	MWRALLOW	M3SYSRSTn

Table 5-85. Flash Registers Memory Map on Control Subsystem

Register Acronym	Register Description	Size (x8)	Type	C28x offset (0x8)	C28x Protection	Reset source
Flash Control Registers				**0x4000**		

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-85. Flash Registers Memory Map on Control Subsystem (continued)

Register Acronym	Register Description	Size (x8)	Type	C28x offset (0x8)	C28x Protection	Reset source
FRDCNTL	Flash Read Control Register	4	R/W	0x0	EALLOW	C28SYSRSTn
FSPRD	Flash read margin control register	4	R/W	0x2	EALLOW	C28SYSRSTn
Reserved	Reserved	52				
FBAC	Flash Bank Access Control Register	4	R/W	0x01E	EALLOW	C28SYSRSTn
FBFALLBACK	Flash Bank Fallback Power Register	4	R/W	0x020	EALLOW	C28SYSRSTn
FBPRDY	Flash Bank Pump Ready Register	4	R	0x022	EALLOW	C28SYSRSTn
FPAC1	Flash Pump Access Control Register 1	4	R/W	0x024	EALLOW	C28SYSRSTn
FPAC2	Flash Pump Access Control Register 2	4	R/W	0x026	EALLOW	C28SYSRSTn
FMAC	Flash Module Access Control Register	4	R	0x028	EALLOW	C28SYSRSTn
FMSTAT	Flash Module Status Register (Used with Flash API – Refer to Flash Application Programming Interface User's Specification for details of this register)	4	R	0x02A	EALLOW	C28SYSRSTn
Reserved	Reserved	680				
FRD_INTF_CTRL	Flash Read Interface Control Register	4	R/W	0x180	EALLOW	C28SYSRSTn
Flash ECC/Error Log Registers				**0x4300**		
ECC_ENABLE	ECC Enable Register	4	R/W	0x0	EALLOW	C28SYSRSTn
SINGLE_ERR_ADDR	Single Error Address Register	4	R	0x2	EALLOW	C28SYSRSTn
UNC_ERR_ADDR	Uncorrectable Error Address Register	4	R	0x4	EALLOW	C28SYSRSTn
ERR_STATUS	Error Status Register	4	R	0x6	EALLOW	C28SYSRSTn
ERR_POS	Error Position Register	4	R	0x8	EALLOW	C28SYSRSTn
ERR_STATUS_CLR	Error Status Clear Register	4	R/W0-1	0xA	EALLOW	C28SYSRSTn
ERR_CNT	Error Counter Register	4	R	0xC	EALLOW	C28SYSRSTn
ERR_THRESHOLD	Error Threshold Register	4	R/W	0xE	EALLOW	C28SYSRSTn
ERR_INTFLG	Error Interrupt Flag Register	4	R	0x10	EALLOW	C28SYSRSTn

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-85. Flash Registers Memory Map on Control Subsystem (continued)

Register Acronym	Register Description	Size (x8)	Type	C28x offset (0x8)	C28x Protection	Reset source
ERR_INTCLR	Error Interrupt Flag Clear Register	4	R/W0-1	0x12	EALLOW	C28SYSRSTn
FDATAH_TEST	Data High Test Register	4	R/W	0x14	EALLOW	C28SYSRSTn
FDATAL_TEST	Data Low Test Register	4	R/W	0x16	EALLOW	C28SYSRSTn
FADDR_TEST	ECC Test Address Register	4	R/W	0x18	EALLOW	C28SYSRSTn
FECC_TEST	ECC Test Register	4	R/W	0x1A	EALLOW	C28SYSRSTn
FECC_CTRL	ECC Control Register	4	R/W	0x1C	EALLOW	C28SYSRSTn
FECC_FOUTH_TEST	Test Data Out High Register	4	R	0x1E	EALLOW	C28SYSRSTn
FECC_FOUTL_TEST	Test Data Out Low Register	4	R	0x20	EALLOW	C28SYSRSTn
FECC_STATUS	ECC Status Register	4	R	0x22	EALLOW	C28SYSRSTn

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.1 Master Subsystem Flash Control Registers

5.4.1.1 Flash Read Control Register (FRDCNTL)

Figure 5-82. Flash Read Control Register (FRDCNTL)

31	12	11	8	7	0
Reserved		RWAIT		Reserved	
R-0		R/W-0xF		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-86. Flash Read Control Register (FRDCNTL) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved		Reserved
11-8	RWAIT		Random read waitstate These bits indicate how many waitstates are added to a flash read access. The RWAIT value can be set anywhere from 0 to 0xF. For a flash access, data is returned in RWAIT+1 M3-SYSCLK cycles. **Note:** The required wait states for each SYSCLK frequency can be found in the device data manual.
7-0	Reserved		Reserved

5.4.1.2 Flash Read Margin Control Register (FSPRD)

Figure 5-83. Flash Read Margin Control Register (FSPRD)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		RM1	RM0
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-87. Flash Read Margin Control Register (FSPRD) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	RM1	0	Read Margin 1 mode is disabled
		1	Read Margin 1 mode is enabled
0	RM0	0	Read Margin 0 mode is disabled
		1	Read Margin 0 mode is enabled

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.1.3 Flash Bank Access Control Register (FBAC)

Figure 5-84. Flash Bank Access Control Register (FBAC)

31	16	15	8	7	0
Reserved		BAGP[7:0]		VREADST	
R-0		R/W-0		R/W-0xF	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-88. Flash Bank Access Control Register (FBAC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-8	BAGP[7:0]		Bank Active Grace Period. These bits contain the starting count value for the BAGP down counter. Any access to a given bank causes its BAGP counter to reload the BAGP value for that bank. After the last access to this flash bank, the down counter delays from 0 to 255 prescaled M3 SYSCLK clock cycles before putting the bank into one of the fallback power modes as determined by the FBFALLBACK register. This value must be greater than 1 when the fallback mode is not ACTIVE. **Note:** The prescaled clock used for the BAGP down counter is a clock divided by 16 from input SYSCLK.
7-0	VREADST		VREAD setup. VREAD is generated by the flash pump and used for flash read operation. The bank power up sequencing starts VREADST HCLK cycles after VREAD power supply becomes stable.

5.4.1.4 Flash Bank Fallback Power Register (FBFALLBACK)

Figure 5-85. Flash Bank Fallback Power Register (FBFALLBACK)

31	2	1	0
Reserved		BNKPWR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-89. Flash Bank Fallback Power Register (FBFALLBACK) Field Description

Bit	Field	Value	Description
31-2	Reserved		Reserved
1-0	BNKPWR		Fall Back power mode
		00	Sleep (Sense amplifiers and sense reference disabled)
		01	Standby (Sense amplifiers disabled, but sense reference enabled)
		10	Reserved
		11	Active (Both sense amplifiers and sense reference enabled)
			Note: If the bank and pump are not in active mode and an access is made, the value of this register is automatically changed to active.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.1.5 Flash Bank Pump Control Register 1 (FBPRDY)

Figure 5-86. Flash Bank Pump Control Register (FBPRDY)

31			16
Reserved			
R-0			

15	14	1	0
PUMPRDY	Reserved		BANKRDY
R-0	R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-90. Flash Bank Pump Control Register (FBPRDY) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	PUMPRDY		Pump Ready. This is a read-only bit which allows software to determine if the pump is ready for flash access before attempting the actual access. If an access is made to a bank when the pump is not ready, wait states are asserted until it becomes ready.
		0	Pump is not ready.
		1	Pump is ready, in active power state.
14-1	Reserved		Reserved
0	BANKRDY		Bank Ready. This is a read-only register which allows software to determine if the M3 bank is ready for Flash access before the access is attempted.
			Note: The user should wait for both the pump and the bank to be ready before attempting an access.
		0	M3 bank is not ready.
		1	M3 bank is in active power mode and is ready for access.

5.4.1.6 Flash Bank Pump Control Register 1 (FPAC1)

Figure 5-87. Flash Bank Pump Control Register 1 (FPAC1)

31	27	26	16
Reserved		PSLEEP	
R-0		R/W-0x64	

15	1	0
Reserved		PMPPWR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-91. Flash Bank Pump Control Register 1 (FPAC1) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26-16	PSLEEP		Pump sleep. These bits contain the starting count value for the charge pump sleep down counter. While the charge pump is in sleep mode, the power mode management logic holds the charge pump sleep counter at this value. When the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP prescaled M3 SYSCLK clock cycles before putting the charge pump into active power mode.
			Note: The pump sleep down counter uses the same prescaled clock as Bank sleep down counter which is divided by 2 of input M3 SYSCLK. For a clock operating at 150 Mhz, the following is the formula used in the obtaining the correct psleep: PSLEEP = PSLEEP_REQ_ns / 2 / HCLK_period_ns PSLEEP = 20000/2/6.66 (in decimal)
15-1	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-91. Flash Bank Pump Control Register 1 (FPAC1) Field Descriptions (continued)

Bit	Field	Value	Description
0	PMPPWR		Flash Charge Pump Fallback Power Mode. This bit selects what power mode the charge pump enters after the pump active grace period (PAGP) counter has timed out.
		0	Sleep (all pump circuits disabled)
		1	Active (all pump circuits active)
			Note: As the pump is shared between both M3 and C28x banks, if an access is made either to the M3 bank or C28x bank, the value of this bit changes to 1 (active).

5.4.1.7 Flash Bank Pump Control Register 2 (FPAC2)

Figure 5-88. Flash Bank Pump Control Register 2 (FPAC2)

31	16	15	0
Reserved		PAGP	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-92. Flash Bank Pump Control Register 2 (FPAC2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	PAGP		Pump Active Grace Period. This register contains the starting count value for the PAGP mode down counter. Any access to flash memory causes the counter to reload with the PAGP value. After the last access to flash memory, the down counter delays from 0 to 65535 prescaled M3 SYSCLK clock cycles before entering one of the charge pump fallback power modes as determined by PUMPPWR in the FPAC1 register. **Note:** The PAGP down counter is clocked by the same prescaled clock as the BAGP down counter which is divided by 16 of input M3 SYSCLK.

5.4.1.8 Flash Module Access Control Register (FMAC)

Figure 5-89. Flash Module Access Control Register (FMAC)

31	3	2	0
Reserved		BANK	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-93. Flash Module Access Control Register (FMAC) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2-0	BANK	0	BANKID. Controls the bank on which the Flash FSM operations will be performed. For these devices, the value of this field will be 0 as there is only one bank in Master subsystem.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.1.9 SECZONEREQUEST Register (SEM) Register

Figure 5-90. SECZONEREQUEST(SEM) Register

31															16
KEY															
R/W-0															
15												3	2	1	0
KEY												Reserved		SEM	
R/W-0												R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-94. SECZONEREQUEST(SEM) Register Field Descriptions

Bit	Field	Value	Description
31-4	KEY		Writing the value 0xC12F59A will allow the writing of the SEM bits, else writes are ignored. Reads will return 0. **Note:** This is to prevent spurious writes of the semaphore bits.
		0	
		1	
3-2	Reserved		Reserved
1-0	SEM		Security Zone Semaphore for M3 Flash Wrapper registers
		00 or 11	M3 Flash Wrapper registers can be written from code running from anywhere without any restriction
		01	M3 Flash Wrapper registers can be written from code running from zone1 security zone
		10	M3 Flash Wrapper registers can be written from code running from zone2 security zone
			State Transitions 00,11 → 01: code running from zone 1 can perform this transition 01 → 00,11: code running from zone 1 can perform this transition 00,11 → 10 : code running from zone 2 can perform this transition 10 → 00,11 : code running from zone 2 can perform this transition Notes: 1. In case SEM = “01” writes and reads to the flash registers (including this register) are blocked if code not in zone1. 2. In case SEM = “10” writes and reads to the flash registers (including this register) are blocked if code not in zone2. 3. In case SEM = “00/11” writes and reads are not blocked. 4. Debug writes to flash registers (including this register) is blocked if SEM = 01 or SEM = 10 In addition, if sectors belonging to zone1 needs to be programmed or erased then SEM should be 01 or zone1 needs to be unlocked. Also, if sectors belonging to zone2 needs to be programmed or erased then SEM should be 10 or zone2 needs to be unlocked.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.1.10 Flash Read Interface Control Register (FRD_INTF_CTRL)

Figure 5-91. Flash Read Interface Control Register (FRD_INTF_CTRL)

31	2	0
Reserved	DATA_CACHE_EN	PROG_CACHE_EN
R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-95. Flash Read Interface Control Register (FRD_INTF_CTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	DATA_CACHE_EN		Data cache enable.
		0	A value of 0 disables the data cache.
		1	A value of 1 enables the data cache.
0	PROG_CACHE_EN		Prefetch enable.
		0	A value of 0 disables program cache and prefetch mechanism.
		1	A value of 1 enables program cache and prefetch mechanism.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.2 Master Subsystem Flash ECC/Error Log Registers

5.4.2.1 ECC Enable Register (ECC_Enable)

Figure 5-92. ECC Enable Register (ECC_Enable)

31	4	3	0
Reserved		ENABLE	
R-0		R/W-0xA	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-96. ECC Enable Register (ECC_Enable) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reserved
3-0	ENABLE	0xA	ECC enable. A value of 0xA would enable ECC. Any other value would disable ECC.

5.4.2.2 Single Error Address Register (SINGLE_ERR_ADDR)

Figure 5-93. Single Error Address Register (SINGLE_ERR_ADDR)

31	0
ERR_ADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-97. Single Error Address Register (SINGLE_ERR_ADDR) Field Descriptions

Bit	Field	Value	Description
31-0	ERR_ADDR		Address at which a single bit error occurred, aligned to a 64-bit boundary.

5.4.2.3 Uncorrectable Error Address Register (UNC_ERR_ADDR)

Figure 5-94. Uncorrectable Error Address Register (UNC_ERR_ADDR)

31	0
UNC_ERR_ADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-98. Uncorrectable Error Address Register (UNC_ERR_ADDR) Field Descriptions

Bit	Field	Value	Description
31-0	UNC_ERR_ADDR		Address at which an un-correctable error occurred, aligned to a 64-bit boundary

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.2.4 Error Status Register (ERR_STATUS)

Figure 5-95. Error Status Register (ERR_STATUS)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		UNC_ERR	FAIL_1	FAIL_0
R-0		R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-99. Error Status Register (ERR_STATUS) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	UNC_ERR		Uncorrectable error. A value of 1 indicates that an un-correctable error occurred. Cleared by writing a 1 to UNC_ERR_CLR bit of ERR_STATUS_CLR register.
1	FAIL_1		Fail on 1.
		0	Fail on 1 single bit error did not occur.
		1	A value of 1 would indicate that a single bit error occurred and the corrected value was 1. Cleared by writing a 1 to FAIL_1_CLR bit of ERR_STATUS_CLR register.
0	FAIL_0		Fail on 0.
		0	Fail on 0 single bit error did not occur.
		1	A value of 1 would indicate that a single bit error occurred and the corrected value was 0. Cleared by writing a 1 to FAIL_0_CLR bit of ERR_STATUS_CLR register.

5.4.2.5 Error Position Register (ERR_POS)

Figure 5-96. Error Position Register (ERR_POS)

31	16
Reserved	
R-0	

15	9	8	7	6	5	0
Reserved		ERR_TYPE	Reserved	ECC_L_OR_H	ERR_POS	
R-0		R-0	R-0	R-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-100. Error Position Register (ERR_POS) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	ERR_TYPE		Error type
		0	Indicates that a single bit error occured in ECC check bits
		1	Indicates that a single bit error occured in data bits
7	Reserved		Reserved
6	ECC_L_OR_H		ECC error location indicator.
		0	Indicates the single-bit error is in lower 64 bits or check bits corresponding to lower 64 bits of data.
		1	Indicates the single-bit error is in upper 64 bits or check bits corresponding to upper 64 bits of data.
5-0	ERR_POS		Error position. Bit position of the single bit error. The position is interpreted depending on whether the ERR_TYPE bit indicates a check bit or a data bit. If ERR_TYPE indicates a check bit error, the error position could range from 0 to 7, else it could range from 0 to 63.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.2.6 Error Status Clear Register (ERR_STATUS_CLR)

Figure 5-97. Error Status Clear Register (ERR_STATUS_CLR)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		UNC_ERR_CLR	FAIL_1_CLR	FAIL_0_CLR
R-0		R/W0-1	R/W0-1	R/W0-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-101. Error Status Clear Register (ERR_STATUS_CLR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	UNC_ERR_CLR		Uncorrectable error clear. Writing a 1 to this bit will clear the UNC_ERR bit of ERR_STATUS register.
1	FAIL_1_CLR		Fail on 1 clear. Writing a 1 to this bit will clear the FAIL_1 bit of ERR_STATUS register. Writes of 0 have no effect.
0	FAIL_0_CLR		Fail on 0 clear. Writing a 1 to this bit will clear the FAIL_0 bit of ERR_STATUS register. Writes of 0 have no effect.

5.4.2.7 Error Counter Register (ERR_CNT)

Figure 5-98. Error Counter Register (ERR_CNT)

31	16	15	0
Reserved		ERR_CNT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-102. Error Counter Register (ERR_CNT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	ERR_CNT		Single bit error count. This counter increments with every single bit ECC error occurrence. Upon reaching the threshold value counter stops counting on single bit errors. ERR_CNT can be cleared (irrespective of whether threshold is met or not) using "Single Err Int Clear" bit. This is applicable for ECC logic test mode and normal operational mode. In ECC logic test mode, ERR_CNT will keep incrementing for every cycle after a single bit error occurrence. So, in order to clear ERR_CNT in ECC logic test mode, ECC logic test mode has to be disabled prior to clearing the ERR_CNT using "Single Err Int Clear" bit.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.2.8 Error Threshold Register (ERR_THRESHOLD)

Figure 5-99. Error Threshold Register (ERR_THRESHOLD)

31	16	15	0
Reserved		THRESHOLD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-103. Error Threshold Register (ERR_THRESHOLD) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	THRESHOLD		Single bit error threshold. Sets the threshold for single bit errors. When the ERR_CNT value equals the THRESHOLD value and a single bit error occurs, SINGLE_ERR_INT flag is set, and an interrupt is fired.

5.4.2.9 Error Interrupt Flag Register (ERR_INTFLG)

Figure 5-100. Error Interrupt Flag Register (ERR_INTFLG)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		UNC_ERR_INT_FLG	SINGLE_ERR_INT_FLG
R-0		R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

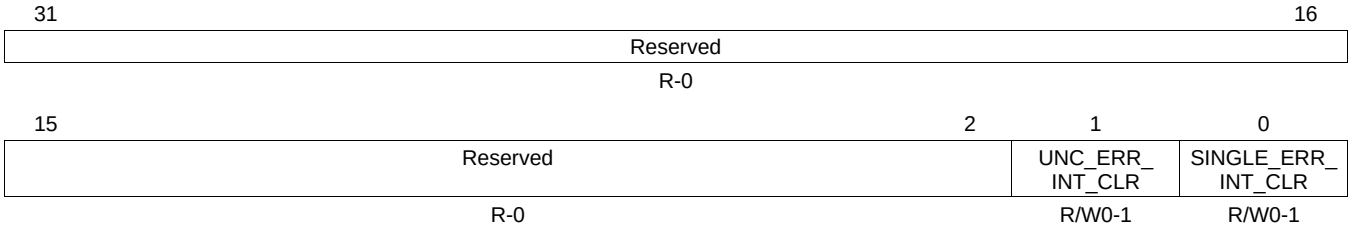
Table 5-104. Error Interrupt Flag Register (ERR_INTFLG) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	UNC_ERR_INT_FLG		Uncorrectable bit error interrupt flag. When a Un-correctable error occurs, this bit is set and the UNC_ERR_INT interrupt is fired. When UNC_ERR_INT_CLR bit of ERR_INTCLR register is written a value of 1 this bit is cleared.
0	SINGLE_ERR_INT_FLG		Single bit error interrupt flag. When the ERR_CNT value equals the ERR_THRESHOLD value and a single bit error occurs then SINGLE_ERR_INT flag is set and SINGLE_ERR_INT interrupt is fired. When SINGLE_ERR_INT_CLR bit of ERR_INTCLR register is written a value of 1 this bit is cleared.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.2.10 Error Interrupt Flag Clear Register (ERR_INTCLR)

Figure 5-101. Error Interrupt Flag Clear Register (ERR_INTCLR)



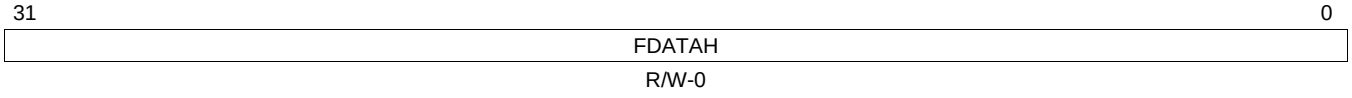
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-105. Error Interrupt Flag Clear Register (ERR_INTCLR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	UNC_ERR_INT_CLR		Uncorrectable bit error interrupt flag clear. Writing a 1 to this bit will clear UNC_ERR_INT_FLG. Writes of 0 have no effect.
0	SINGLE_ERR_INT_CLR		Single bit error interrupt flag clear. Writing a 1 to this bit will clear SINGLE_ERR_INT_FLG. Writes of 0 have no effect.

5.4.2.11 Data High Test Register (FDATAH_TEST)

Figure 5-102. Data High Test Register (FDATAH_TEST)



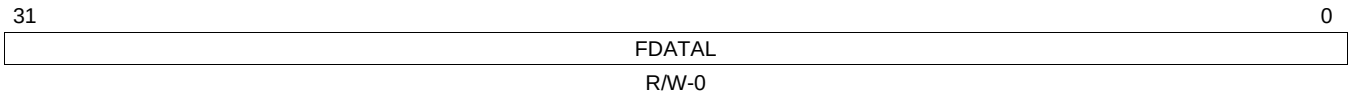
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-106. Data High Test Register (FDATAH_TEST) Field Descriptions

Bit	Field	Value	Description
31-0	FDATAH		High double word of selected 64-bit data. User-configurable bits 63:32 of the selected data blocks in ECC test mode.

5.4.2.12 Data Low Test Register (FDATAL_TEST)

Figure 5-103. Data Low Test Register (FDATAL_TEST)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-107. Data Low Test Register (FDATAL_TEST) Field Descriptions

Bit	Field	Value	Description
31-0	FDATAL		Low double word of selected 64-bit data. User-configurable bits 31:0 of the selected data blocks in ECC test mode.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.2.13 ECC Test Address Register (FADDR_TEST)

Figure 5-104. ECC Test Address Register (FADDR_TEST)

31	19	18	0
Reserved		ADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-108. ECC Test Address Register (FADDR_TEST) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved		Reserved
18-0	ADDR		Address for selected 64-bit data. User-configurable address bits of the selected data in ECC test mode.

5.4.2.14 ECC Test Register (FECC_TEST)

Figure 5-105. ECC Test Register (FECC_TEST)

31	8	7	0
Reserved		ECC	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-109. ECC Test Register (FECC_TEST) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	ECC		8-bit ECC for selected 64-bit data. User-configurable ECC bits of the selected 64-bit data block in ECC test mode.

5.4.2.15 ECC Control Register (FECC_CTRL)

Figure 5-106. ECC Control Register (FECC_CTRL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		ECC_SELECT	ECC_TEST_EN
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-110. ECC Control Register (FECC_CTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	ECC_SELECT		ECC block select.
		0	Selects the ECC block on bits [63:0] of bank data.
		1	Selects the ECC block on bits [127:64] of bank data.
0	ECC_TEST_EN		ECC test mode enable.
		0	ECC test mode disabled
		1	ECC test mode enabled

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.2.16 Test Data Out High Register (FECC_FOUTH_TEST)

Figure 5-107. Test Data Out High Register (FECC_FOUTH_TEST)

31	0
DATAOUTH	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-111. Test Data Out High Register (FECC_FOUTH_TEST) Field Descriptions

Bit	Field	Value	Description
31-0	DATAOUTH		High double word test data out. Holds bits 63:32 of the data out of the selected ECC block.

5.4.2.17 Test Data Out Low Register (FECC_FOUTL_TEST)

Figure 5-108. Test Data Out Low Register (FECC_FOUTL_TEST)

31	0
DATAOUTL	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-112. Test Data Out Low Register (FECC_FOUTL_TEST) Field Descriptions

Bit	Field	Value	Description
31-0	DATAOUTL		Low double word test data out. Holds bits 31:0 of the data out of the selected ECC block.

5.4.2.18 ECC Status Register (FECC_STATUS)

Figure 5-109. ECC Status Register (FECC_STATUS)

31	16
Reserved	
R-0	

15	9	8	7	2	1	0
Reserved		CHK_ERR	ERR_POS		UNC_ERR	SINGLE_ERR
R-0		R-0	R-0		R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-113. ECC Status Register (FECC_STATUS) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	CHK_ERR		Test mode ECC single bit error indicator. When 1, indicates that the single bit error is in check bits. When 0, indicates that the single bit error is in data bits (If SINGLE_ERR field is also set.
7-2	ERR_POS		Test mode single bit error position. Holds the bit position where the single bit error occurred. The position is interpreted depending on whether the CHK_ERR bit indicates a check bit or a data bit. If CHK_ERR indicates a check bit error, the error position could range from 0 to 7, or it could range from 0 to 63.
1	UNC_ERR		Test mode ECC double bit error. When 1 indicates that the ECC test resulted in an uncorrectable bit error.
0	SINGLE_ERR		Test mode ECC single bit error. When 1 indicates that the ECC test resulted in a single bit error.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.3 Control Subsystem Flash Control Registers

5.4.3.1 Flash Read Control Register (FRDCNTL)

Figure 5-110. Flash Read Control Register (FRDCNTL)

31	11	12	8	7	0
Reserved		RWAIT		Reserved	
R-0		R/W-0xF		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-114. Flash Read Control Register (FRDCNTL) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved		Reserved
11-8	RWAIT		Random read waitstate These bits indicate how many waitstates are added to a flash read access. The RWAIT value can be set anywhere from 0 to 0xF. For a flash access, data is returned in RWAIT+1 C28 SYSCLK cycles. **Note:** The required wait states for each SYSCLK frequency can be found in the device data manual.
7-0	Reserved		Reserved

5.4.3.2 Flash Read Margin Control Register (FSPRD)

Figure 5-111. Flash Read Margin Control Register (FSPRD)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		RM1	RM0
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-115. Flash Read Margin Control Register (FSPRD) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	RM1	0	Read Margin 1 mode is disabled
		1	Read Margin 1 mode is enabled
0	RM0	0	Read Margin 0 mode is disabled
		1	Read Margin 0 mode is enabled

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.3.3 Flash Bank Access Control Register (FBAC)

Figure 5-112. Flash Bank Access Control Register (FBAC)

31	16	15	8	7	0
Reserved		BAGP[7:0]		VREADST	
R-0		R/W-0		R/W-0xF	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-116. Flash Bank Access Control Register (FBAC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-8	BAGP[7:0]		Bank Active Grace Period. These bits contain the starting count value for the BAGP down counter. Any access to a given bank causes its BAGP counter to reload the BAGP value for that bank. After the last access to this flash bank, the down counter delays from 0 to 255 prescaled C28x SYSCLK clock cycles before putting the bank into one of the fallback power modes as determined by the FBFALLBACK register. This value must be greater than 1 when the fallback mode is not ACTIVE. **Note:** The prescaled clock used for the BAGP down counter is a clock divided by 16 from input C28x SYSCLK.
7-0	VREADST		VREAD setup. VREAD is generated by the flash pump and used for flash read operation. The bank power up sequencing starts VREADST HCLK cycles after VREAD power supply becomes stable.

5.4.3.4 Flash Bank Fallback Power Register (FBFALLBACK)

Figure 5-113. Flash Bank Fallback Power Register (FBFALLBACK)

31	2	1	0
Reserved		BNKPWR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-117. Flash Bank Fallback Power Register (FBFALLBACK) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1-0	BNKPWR		Fall back power mode.
		00	Sleep (Sense amplifiers and sense reference disabled)
		01	Standby (Sense amplifiers disabled, but sense reference enabled)
		10	Reserved
		11	Active (Both sense amplifiers and sense reference enabled)
			Note: If the bank and pump are not in active mode and an access is made, the value of this register is automatically changed to active.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.3.5 Flash Bank Pump Control Register 1 (FBPRDY)

Figure 5-114. Flash Bank Pump Control Register (FBPRDY)

31	16
Reserved	
R-0	

15	14	1	0
PUMPRDY	Reserve		BANKRDY
R-0	R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-118. Flash Bank Pump Control Register (FBPRDY) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	PUMPRDY		Pump Ready. Pump Ready is a read-only bit which allows software to determine if the pump is ready for flash access before attempting the actual access. If an access is made to a bank when the pump is not ready, wait states are asserted until it becomes ready.
		0	Pump is not ready.
		1	Pump is ready, in active power state.
14-1	Reserved		Reserved
0	BANKRDY		Bank Ready. This is a read-only register which allows software to determine if the C28x bank is ready for Flash access before the access is attempted. **Note:** User should wait for both the pump and the bank to be ready before attempting an access.
		0	C28x bank is not ready.
		1	C28x bank is in active power mode and is ready for access.

5.4.3.6 Flash Bank Pump Control Register 1 (FPAC1)

Figure 5-115. Flash Bank Pump Control Register 1 (FPAC1)

31	27	26	16
Reserved		PSLEEP	
R-0		R/W-0x64	

15	1	0
Reserved		PMPPWR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-119. Flash Bank Pump Control Register 1 (FPAC1) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reserved
26-16	PSLEEP		Pump sleep. These bits contain the starting count value for the charge pump sleep down counter. While the charge pump is in sleep mode, the power mode management logic holds the charge pump sleep counter at this value. When the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP prescaled SYSCLK clock cycles before putting the charge pump into active power mode. **Note:** The pump sleep down counter uses the same prescaled clock as Bank sleep down counter which is divided by 2 of input SYSCLK.
15-1	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-119. Flash Bank Pump Control Register 1 (FPAC1) Field Descriptions (continued)

Bit	Field	Value	Description
0	PMPPWR		Flash Charge Pump Fallback Power Mode. This bit selects what power mode the charge pump enters after the pump active grace period (PAGP) counter has timed out.
		0	Sleep (all pump circuits disabled)
		1	Active (all pump circuits active)
			Note: As the pump is shared between both M3 and C28x banks, if an access is made either to the M3 bank or C28x bank, the value of this bit changes to 1 (active).

5.4.3.7 Flash Bank Pump Control Register 2 (FPAC2)

Figure 5-116. Flash Bank Pump Control Register 2 (FPAC2)

31	16	15	0
Reserved		PAGP	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-120. Flash Bank Pump Control Register 2 (FPAC2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	PAGP		Pump Active Grace Period. This register contains the starting count value for the PAGP mode down counter. Any access to flash memory causes the counter to reload with the PAGP value. After the last access to flash memory, the down counter delays from 0 to 65535 prescaled C28x SYSCLK clock cycles before entering one of the charge pump fallback power modes as determined by PUMPPWR in the FPAC1 register. **Note:** The PAGP down counter is clocked by the same prescaled clock as the BAGP down counter which is divided by 16 of input C28x SYSCLK.

5.4.3.8 Flash Module Access Control Register (FMAC)

Figure 5-117. Flash Module Access Control Register (FMAC)

31	3	2	0
Reserved		BANK	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-121. Flash Module Access Control Register (FMAC) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2-0	BANK		Bank ID. Controls the bank on which the Flash FSM operations will be performed. For these devices, the value of this field will be 0 as there is only one bank in the control subsystem

5.4.3.9 Flash Read Interface Control Register (FRD_INTF_CTRL)

Figure 5-118. Flash Read Interface Control Register (FRD_INTF_CTRL)

31	2	1	0
Reserved		DATA_CACHE_EN	PROG_CACHE_EN
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-122. Flash Read Interface Control Register (FRD_INTF_CTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	DATA_CACHE_EN		Data cache enable.
		0	A value of 0 disables the data cache.
		1	A value of 1 enables the data cache.
0	PROG_CACHE_EN		Prefetch enable.
		0	A value of 0 disables program cache and prefetch mechanism.
		1	A value of 1 enables program cache and prefetch mechanism.

5.4.4 Control Subsystem Flash ECC/Error Log Registers

5.4.4.1 ECC Enable Register (ECC_ENABLE)

Figure 5-119. ECC Enable Register (ECC_ENABLE)

31	4	3	0
Reserved		ENABLE	
R-0		R/W-0xA	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-123. ECC Enable Register (ECC_ENABLE) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3-0	ENABLE		ECC enable. A value of 0xA would enable ECC. Any other value would disable ECC.

5.4.4.2 SIngle Error Address Register (SINGLE_ERR_ADDR)

Figure 5-120. SIngle Error Address Register (SINGLE_ERR_ADDR)

31	0
ERR_ADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-124. SIngle Error Address Register (SINGLE_ERR_ADDR) Field Descriptions

Bit	Field	Value	Description
31-0	ERR_ADDR		Single bit error address. Address at which a single bit error occurred, aligned to a 128 bit boundary.

5.4.4.3 Uncorrectable Error Address Register (UNC_ERR_ADDR)

Figure 5-121. Uncorrectable Error Address Register (UNC_ERR_ADDR)

31	0
UNC_ERR_ADDR	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-125. Uncorrectable Error Address Register (UNC_ERR_ADDR) Field Descriptions

Bit	Field	Value	Description
31-0	UNC_ERR_ADDR		Uncorrectable error address. Address at which un-correctable error occurred, aligned to a 128 bit boundary.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.4.4 Error Status Register (ERR_STATUS)

Figure 5-122. Error Status Register (ERR_STATUS)

31	3	2	1	0
Reserved		UNC_ERR	FAIL_1	FAIL_0
R-0		R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-126. Error Status Register (ERR_STATUS) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	UNC_ERR		Uncorrectable error. A value of 1 indicates that an un-correctable error occurred. Cleared by writing a 1 to UNC_ERR_CLR bit of ERR_STATUS_CLR register.
1	FAIL_1		Fail on 1
		0	Fail on 1 single bit error did not occur.
		1	A value of 1 would indicate that a single bit error occurred and the corrected value was 1. Cleared by writing a 1 to FAIL_1_CLR bit of ERR_STATUS_CLR register.
0	FAIL_0		Fail on 0
		0	Fail on 0 single bit error did not occur.
		1	A value of 1 would indicate that a single bit error occurred and the corrected value was 0. Cleared by writing a 1 to FAIL_0_CLR bit of ERR_STATUS_CLR register

5.4.4.5 Error Position Register (ERR_POS)

Figure 5-123. Error Position Register (ERR_POS)

31	16
Reserved	
R-0	

15	9	8	7	6	5	0
Reserved		ERR_TYPE	Reserved	ECC_L_OR_H	ERR_POS	
R-0		R-0	R-0	R-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

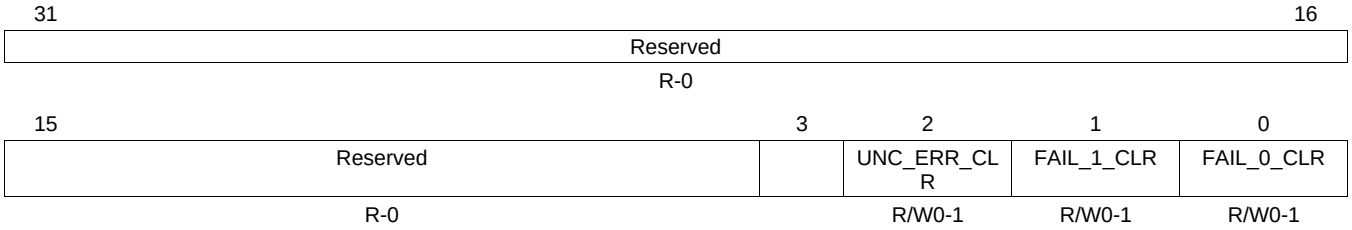
Table 5-127. Error Position Register (ERR_POS) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	ERR_TYPE		Error type
		0	Indicates that a single bit error occured in ECC check bits
		1	Indicates that a single bit error occured in data bits
7	Reserved		Reserved
6	ECC_L_OR_H		ECC error location indicator.
		0	Indicates the single-bit error is in lower 64 bits or check bits corresponding to lower 64 bits of data.
		1	Indicates the single-bit error is in upper 64 bits or check bits corresponding to upper 64 bits of data.
5-0	ERR_POS		Error position. Bit position of the single bit error. The position is interpreted depending on whether ERR_TYPE bit indicates a check bit or a data bit. If ERR_TYPE indicates a check bit error, the error position could range from 0 to 7 else it could range from 0 to 63.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.4.6 Error Status Clear Register (ERR_STATUS_CLR)

Figure 5-124. Error Status Clear Register (ERR_STATUS_CLR)



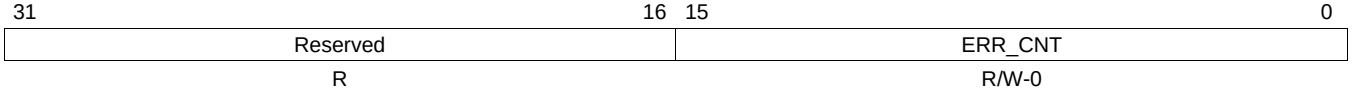
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-128. Error Status Clear Register (ERR_STATUS_CLR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	UNC_ERR_CLR		Uncorrectable error clear. Writing a 1 to this bit will clear the UNC_ERR bit of ERR_STATUS register.
1	FAIL_1_CLR		Fail on 1 clear. Writing a 1 to this bit will clear the FAIL_1 bit of ERR_STATUS register. Writes of 0 have no effect.
0	FAIL_0_CLR		Fail on 0 clear. Writing a 1 to this bit will clear the FAIL_0 bit of ERR_STATUS register. Writes of 0 have no effect.

5.4.4.7 Error Counter Register (ERR_CNT)

Figure 5-125. Error Counter Register (ERR_CNT)



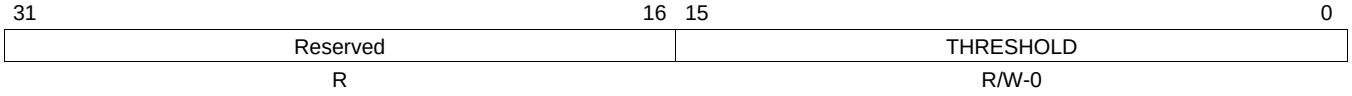
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-129. Error Counter Register (ERR_CNT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	ERR_CNT		Single bit error count. This counter increments with every single bit ECC error occurrence. Upon reaching the threshold value counter stops counting on single bit errors. ERR_CNT can be cleared (irrespective of whether threshold is met or not) using "Single Err Int Clear" bit. This is applicable for ECC logic test mode and normal operational mode. In ECC logic test mode, ERR_CNT will keep incrementing for every cycle after a single bit error occurrence. So, in order to clear ERR_CNT in ECC logic test mode, ECC logic test mode has to be disabled prior to clearing the ERR_CNT using "Single Err Int Clear" bit.

5.4.4.8 Error Threshold Register (ERR_THRESHOLD)

Figure 5-126. Error Threshold Register (ERR_THRESHOLD)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-130. Error Threshold Register (ERR_THRESHOLD) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	THRESHOLD		Single bit error threshold. Sets the threshold for single bit errors. When the ERR_CNT value equals the THRESHOLD value and a single bit error occurs, SINGLE_ERR_INT flag is set, and an interrupt is fired.

5.4.4.9 Error Interrupt Flag Register (ERR_INTFLG)

Figure 5-127. Error Interrupt Flag Register (ERR_INTFLG)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		UNC_ERR_INT_FLG	SINGLE_ERR_INT_FLG
R-0		R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-131. Error Interrupt Flag Register (ERR_INTFLG) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	UNC_ERR_INT_FLG		Uncorrectable bit error interrupt flag. When a Un-correctable error occurs, this bit is set and the UNC_ERR_INT interrupt is fired. When UNC_ERR_INT_CLR bit of ERR_INTCLR register is written a value of 1 this bit is cleared.
0	SINGLE_ERR_INT_FLG		Single bit error interrupt flag. When the ERR_CNT value equals the ERR_THRESHOLD value and a single bit error occurs then SINGLE_ERR_INT flag is set and SINGLE_ERR_INT interrupt is fired. When SINGLE_ERR_INT_CLR bit of ERR_INTCLR register is written a value of 1 this bit is cleared.

5.4.4.10 Error Interrupt Flag Clear Register (ERR_INTCLR)

Figure 5-128. Error Interrupt Flag Clear Register (ERR_INTCLR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		UNC_ERR_INT_CLR	SINGLE_ERR_INT_CLR
R-0		R/W0-1	R/W0-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

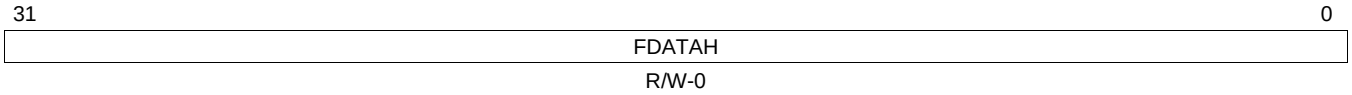
Table 5-132. Error Interrupt Flag Clear Register (ERR_INTCLR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	UNC_ERR_INT_CLR		Uncorrectable bit error interrupt flag clear. Writing a 1 to this bit will clear UNC_ERR_INT_FLG. Writes of 0 have no effect.
0	SINGLE_ERR_INT_CLR		Single bit error interrupt flag clear. Writing a 1 to this bit will clear SINGLE_ERR_INT_FLG. Writes of 0 have no effect.

Copyright © 2012–2019, Texas Instruments Incorporated

5.4.4.11 Data High Test Register (FDATAH_TEST)

Figure 5-129. Data High Test Register (FDATAH_TEST)



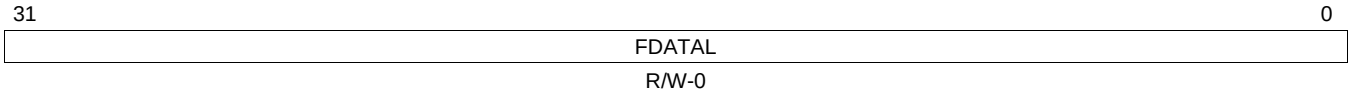
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-133. Data High Test Register (FDATAH_TEST) Field Descriptions

Bit	Field	Value	Description
31-0	FDATAH		High double word of selected 64-bit data. User-configurable bits 63:32 of the selected data blocks in ECC test mode.

5.4.4.12 Data Low Test Register (FDATAL_TEST)

Figure 5-130. Data Low Test Register (FDATAL_TEST)



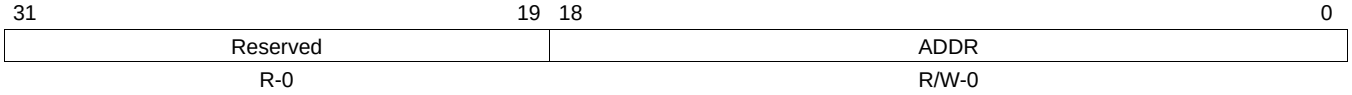
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-134. Data Low Test Register (FDATAL_TEST) Field Descriptions

Bit	Field	Value	Description
31-0	FDATAL		Low double word of selected 64-bit data. User-configurable bits 31:0 of the selected data blocks in ECC test mode.

5.4.4.13 ECC Test Address Register (FADDR_TEST)

Figure 5-131. ECC Test Address Register (FADDR_TEST)



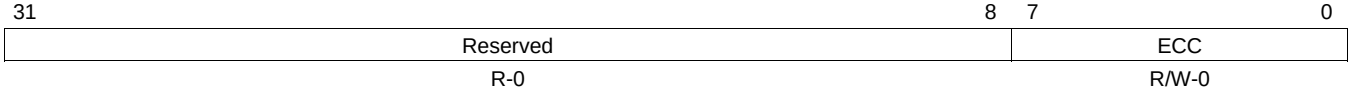
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-135. ECC Test Address Register (FADDR_TEST) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved		Reserved
18-0	ADDR		Address for selected 64-bit data. User-configurable address bits of the selected data in ECC test mode.

5.4.4.14 ECC Test Register (FECC_TEST)

Figure 5-132. ECC Test Register (FECC_TEST)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-136. ECC Test Register (FECC_TEST) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	ECC		8-bit ECC for selected 64-bit data. User-configurable ECC bits of the selected 64-bit data block in ECC test mode.

5.4.4.15 ECC Control Register (FECC_CTRL)

Figure 5-133. ECC Control Register (FECC_CTRL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		ECC_SELECT	ECC_TEST_EN
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-137. ECC Control Register (FECC_CTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	ECC_SELECT		ECC block select.
		0	Selects the ECC block on bits [63:0] of bank data.
		1	Selects the ECC block on bits [127:64] of bank data.
0	ECC_TEST_EN		ECC test mode enable.
		0	ECC test mode disabled
		1	ECC test mode enabled

5.4.4.16 Test Data Out High Register (FECC_FOUTH_TEST)

Figure 5-134. Test Data Out High Register (FECC_FOUTH_TEST)

31	0
DATAOUTH	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-138. Test Data Out High Register (FECC_FOUTH_TEST) Field Descriptions

Bit	Field	Value	Description
31-0	DATAOUTH		High double word test data out. Holds bits 63:32 of the data out of the selected ECC block.

5.4.4.17 Test Data Out Low Register (FECC_FOUTL_TEST)

Figure 5-135. Test Data Out Low Register (FECC_FOUTL_TEST)

31	0
DATAOUTL	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 5-139. Test Data Out Low Register (FECC_FOUTL_TEST) Field Descriptions

Bit	Field	Value	Description
31-0	DATAOUTL		Low double word test data out. Holds bits 31:0 of the data out of the selected ECC block.

5.4.4.18 ECC Status Register (FECC_STATUS)

Figure 5-136. ECC Status Register (FECC_STATUS)

31	16
Reserved	
R-0	

15	11	10	8	7	2	1	0
Reserved		CHK_ERR_POS		DATA_ERR_POS		UNC_ERR	SINGLE_ERR
R-0		R-0		R-0		R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5-140. ECC Status Register (FECC_STATUS) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved		Reserved
10-8	CHK_ERR_POS		Test mode ECC bit error position. Holds the bit position in the 8 check bits where the error occurred.
7-2	DATA_ERR_POS		Test mode data bit error position. Holds the bit position in the 64 bit data where the error occurred.
1	UNC_ERR		Test mode ECC double bit error. When 1 indicates that the ECC test resulted in an uncorrectable bit error.
0	SINGLE_ERR		Test mode ECC single bit error. When 1 indicates that the ECC test resulted in a single bit error.

Copyright © 2012–2019, Texas Instruments Incorporated

ROM Code and Peripheral Booting

This chapter describes the booting functionality of the master (M3) and control (C28) subsystems.

Topic **Page**

6.1 Introduction 537
6.2 Device Boot Sequence 537
6.3 Device Boot Modes 538
6.4 Device Boot Flow Diagram 540
6.5 M-Boot ROM Description 541
6.6 C-Boot ROM Description 570
6.7 Guidelines for Boot ROM Application Writers 614
6.8 Application Notes to Use Boot ROM 619

Copyright © 2012–2019, Texas Instruments Incorporated

6.1 Introduction

The master subsystem and control subsystem, each has an independent boot ROM associated with it, as shown in the memory map. Each ROM has a bootloader programmed in it by TI which is executed when the device is powered ON, and each time the device is reset. The bootloader is used as an initial program to load application on to device RAM through any of the bootable peripherals or configured to start application in flash if any.

The master subsystem has 64KB of boot ROM, which will be referred to as M-Boot ROM. The control subsystem has 64 KB of boot ROM which will be referred to as C-Boot ROM. The M3 CPU acts as the master device and can control the booting process of C28x, so M-Boot ROM is also referred to as master boot ROM and C-Boot ROM is referred to as the control subsystem boot ROM. As explained in the Resets section of the *System Control and Interrupts* chapter, whenever the device is powered up or reset by an external reset, the master subsystem – the Cortex M3 CPU will begin executing code from M-Boot ROM. As will be explained in this chapter master Boot ROM will bring the control subsystem out of reset and C28x PU will begin executing code in the control subsystem Boot ROM.

This chapter explains the boot procedure followed by both M-Boot ROM and C-Boot ROM, when executed and how it enables users to load application code on to device or start a user application in flash.

6.2 Device Boot Sequence

The boot sequence is described below:

- The factory default bootloader runs on each subsystem and initializes the system so that it will be able to download user application through one of the booting peripherals on to device RAMs or starts the application in respective flash memories.
- The control subsystem is held in reset on power up and M-Boot ROM brings the control subsystem out of reset.
- C-Boot ROM, after performing the needed device initialization, puts the C28x CPU core in IDLE mode.
- C-Boot ROM supports various IPC commands to allow the master subsystem application to control how C-Boot ROM should boot.
- When the device is set to boot from flash via the GPIO boot pins, M-Boot ROM brings the control subsystem out of reset so that C-Boot ROM can run and branches to the master subsystem flash application start address.
- When the device is set to boot from RAM, M-Boot ROM brings the control subsystem out of reset and branches to the master subsystem RAM start address.
- When the device is set to boot from any of the bootable master subsystem peripherals, M-Boot ROM will bring the control subsystem out of reset and looks for boot commands/data from the respective master subsystem peripherals.
- C-Boot ROM does not start its application until/unless commanded to do so by the master subsystem application using an IPC command, even when there is a valid application present in C-Flash.
- M-Boot ROM only allows C-Boot ROM to start; it does not dictate how C-Boot ROM should boot the control subsystem. It is up to the master subsystem application to control how the control subsystem should boot. The master subsystem application can do so by using any of the C-Boot ROM supported IPC commands.
- Booting protocols are explained below for both M-Boot ROM and C-Boot ROM. On C-Boot ROM, this device supports legacy C28x-style booting and on M-Boot ROM, this device supports LM3S9B96-style booting.
- It is possible for the master subsystem application to boot the control subsystem using one of the methods below :
  - Load the application that the control subsystem has to RUN on to shared RAM, and use IPC command to ask C-Boot ROM to branch to the shared RAM location.
  - Use one of the available IPC commands to ask C-Boot ROM to boot from the control subsystem flash.
  - Use one of the available IPC commands to ask C-Boot ROM to boot from the control subsystem peripherals.

Copyright © 2012–2019, Texas Instruments Incorporated

6.3 Device Boot Modes

This section explains the boot mode(s) supported on this device. As part of the master subsystem, M-Boot ROM reads the boot mode GPIO on startup and boots the device accordingly. Table 6-1 shows the boot mode GPIO configuration supported.

As will be explained in this chapter, C-Boot ROM also supports different boot modes which allow it to boot from the control subsystem peripherals or boot-to-flash or boot-to-RAM. It is up to the master subsystem application to start these boot mode(s) in C-Boot ROM using IPC commands.

Table 6-1. Master Subsystem Boot Mode Selection(1)

BOOT MODE NO.	MASTER SUBSYSTEM BOOT MODES	PF2_GPIO34 (Bmode_pin4)(2)	PF3_GPIO35 (Bmode_pin3)(2)	PG7_GPIO47 (Bmode_pin2)(2)	PG3_GPIO43 (Bmode_pin1)(2)
0	Boot from Parallel GPIO	0	0	0	0
1	Boot to Master Subsystem RAM	0	0	0	1
2	Boot from Master Subsystem serial peripherals (UART0/SSI0/I2C0)	0	0	1	0
3	Boot from Master Subsystem CAN interface	0	0	1	1
4	Boot from Master Subsystem Ethernet interface	0	1	0	0
5	Not supported (Defaults to Boot-to-Flash)	0	1	0	1
6(3)	Boot-to-OTP	0	1	1	0
7	Boot to Master Subsystem Flash memory	0	1	1	1
8(4)	Not supported (Defaults to Boot-to-Flash)	1	0	0	0
9(4)	Boot from Master Subsystem serial peripheral – SSI0 Master	1	0	0	1
10(4)	Boot from Master Subsystem serial peripheral – I2C0 Master	1	0	1	0
11(4)	Not supported (Defaults to Boot-to-Flash)	1	0	1	1
12(4)	Not supported (Defaults to Boot-to-Flash)	1	1	0	0
13(4)	Not supported (Defaults to Boot-to-Flash)	1	1	0	1
14(4)	Not supported (Defaults to Boot-to-Flash)	1	1	1	0
15(4)	Not supported (Defaults to Boot-to-Flash)	1	1	1	1

(1) Silicon revision A allows the user to change the GPIO pins used to determine the boot mode. Silicon revision 0 does not have this option.

(2) By default, GPIO terminals are not pulled up (they are floating).

(3) On silicon revision 0, PF2_GPIO34 is a "don't care". So, the state of PF2_GPIO34 should not affect boot mode selection.

(4) Boot Modes 8–15 are not supported on silicon revision 0.

(5) Supported only in TMS version. On all other versions, this mode defaults to Boot-to-Flash.

It is advised that users should not use the unsupported boot mode configuration to let the device boot-to-flash by default, in case TI adds new peripheral boot mode(s) in subsequent revisions of Boot ROM.

6.3.1 Configuring Alternate Boot Mode Pins

NOTE: This feature is NOT available in the Sonata Rev0 silicon.

This device offers the flexibility to select alternate boot mode pins if the factory-offered default boot mode pins are not convenient for customer design. If the user wants to change any of the four default boot mode pins, a 32-bit OTP location (0x680824) can be programmed in the format described below. This location, called “OTP BOOTMODE GPIO CONFIG REGISTER,” is fragmented into four 8-bit fields named bmode_pin1 (bits 0:7), bmode_pin2 (bits 8:15), Bmode_pin3 (bits 16:23), and bmode_pin4 (bits 24:31).

OTP BOOTMODE GPIO CONFIG REGISTER at 0x680824 location.

Copyright © 2012–2019, Texas Instruments Incorporated

Bits 24:31	Bits 16:23	Bits 8:15	Bits 0:7
Bmode_pin4	Bmode_pin3	Bmode_pin2	Bmode_pin1

To change the default boot mode pin to GPIOx, program GPIOx+1 in the corresponding bit field.

Bmode_pin1 (bits 0:7)	The default pin1 boot mode pin is PG3_GPIO43. For example, if the user wants to choose PA2_GPIO2, then program bits 0:7 with 0x3 (decimal 3).
Bmode_pin2 (bits 8:15)	The default pin2 boot mode pin is PG7_GPIO47. For example, if the user wants to choose PF2_GPIO34, then program bits 8:15 with 0x23(decimal 35).
Bmode_pin3 (bits 16:23)	The default pin3 boot mode pin is PF3_GPIO35. For example, if the user wants to use the same default pin3 boot mode pin (PF3_GPIO35), then leave bits 16:23 unprogrammed.
Bmode_pin4 (bits 24:31)	The default pin4 boot mode pin is PF2_GPIO34. For example, if the user wants to choose PC5_GPIO69, then program bits 24:31 with 0x46 (decimal 70).

CAUTION

Once programmed, the OTP BOOTMODE GPIO CONFIG register CANNOT be erased. So, program this location ONLY if you have finalized the GPIO.

Restrictions:

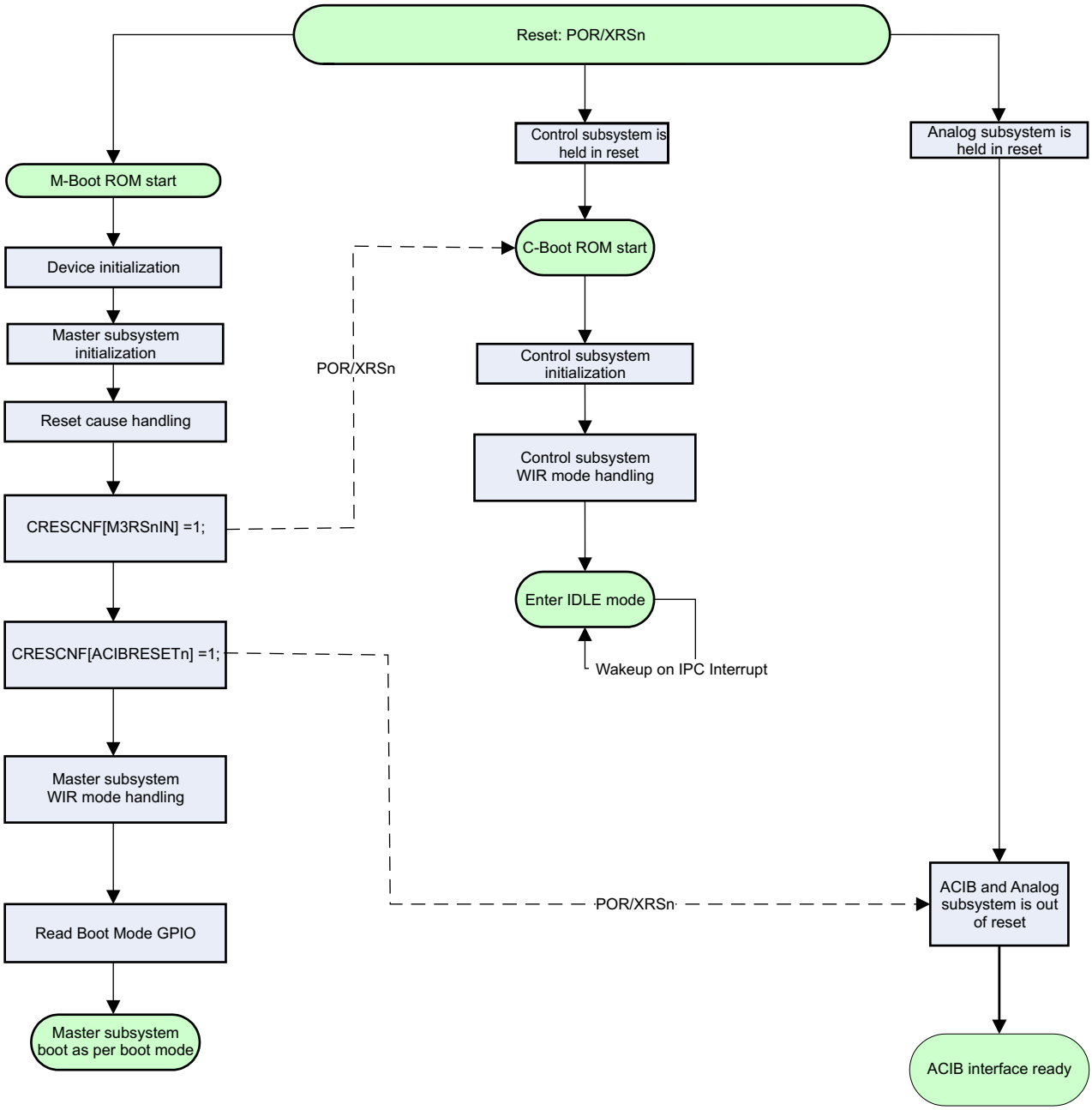
- The chosen boot mode pins should be between GPIO0 and GPIO71. In case of any illegal GPIO entry, boot ROM defaults to factory-selected boot mode pins.
- The following GPIOs (GPIO39, GPIO44, GPIO64, GPIO65, GPIO66, GPIO67) cannot be used as boot mode pins, as these pins are not pinned out on this device. Selecting these pins as boot mode pins is considered an illegal GPIO entry.

Copyright © 2012–2019, Texas Instruments Incorporated

6.4 Device Boot Flow Diagram

Figure 6-1 shows the device boot flow for the master, control, and analog subsystems on power-up or after an external reset input.

Figure 6-1. Device Boot Flow



Copyright © 2012–2019, Texas Instruments Incorporated

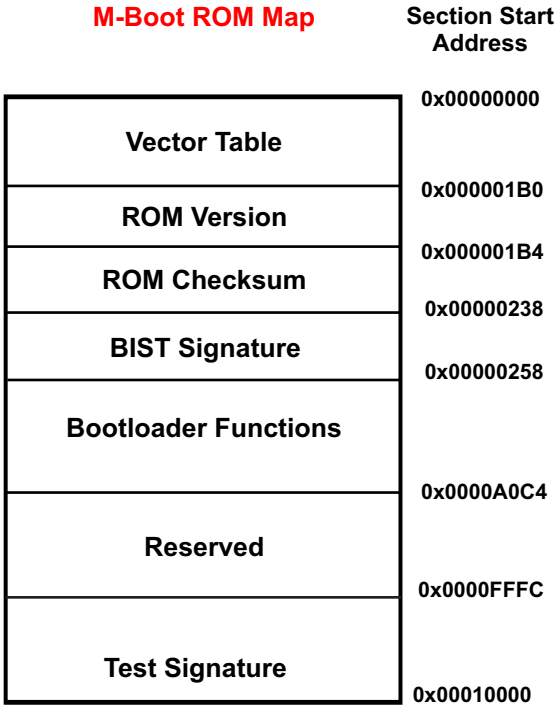
6.5 M-Boot ROM Description

This section explains the bootloader in the master subsystem ROM.

6.5.1 M-Boot ROM Memory Map

On these devices, ROM is mapped beginning at the 0x00000000 address. So whenever the Cortex M3 CPU is reset, it will fetch the reset vector from the M-Boot ROM Vector table and the CPU starts executing code in M-Boot ROM. Figure 6-2 is the memory map of M-Boot ROM.

Figure 6-2. M-Boot ROM Memory Map



Note: On this device, M-Boot ROM is dual-mapped at the beginning of 0x00000000 and 0x00010000.

6.5.2 M-Boot ROM Vector Table

The Cortex-M3 CPU vector table on M-Boot ROM resides at addresses from 0x00000000 to 0x000001B0. On reset, Cortex-M3 NVIC fetches the stack pointer from the first location in this table and the reset vector is fetched from address 0x00000004 in the table. As long as boot ROM is executing, the NVIC base address is set to its default value (0x00000000) and boot ROM does not move this base address. It is up to the user's application to move the NVIC base address as per the application's needs. So any exception or interrupt occurring during boot is handled by the handlers registered in this boot ROM vector table.

On this device all NMIs are enabled by default, so it is necessary for M-Boot ROM to handle all the NMIs that might occur during boot process. Refer to Section 6.5.14 for more details on how M-Boot ROM handles different interrupts and exceptions that occur during boot.

Table 6-2. M-Boot ROM Vector Table

Vector Name (Number)	Vector Address or Location in Boot ROM	Contents (Handler address)
Stack Pointer at reset (0)	0x00000000	0x20004900
Reset (1)	0x00000004	ResetISR
Non-Maskable Interrupt(2)	0x00000008	mbrom_nmi_interrupt_handler
Hard Fault (3)	0x0000000C	mbrom_hard_fault_isr_handler
Memory Management (4)	0x00000010	IntDefaultHandler

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-2. M-Boot ROM Vector Table (continued)

Vector Name (Number)	Vector Address or Location in Boot ROM	Contents (Handler address)
Bus Fault (5)	0x00000014	IntDefaultHandler
Usage Fault (6)	0x00000018	IntDefaultHandler
Reserved (7-10)	0x0000001C – 0x00000028	Reserved (0x00000000)
SVCall (11)	0x0000002C	IntDefaultHandler
DebugMonitor(12)	0x00000030	IntDefaultHandler
Reserved (13)	0x00000034	Reserved (0x00000000)
PendSV (14)	0x00000038	IntDefaultHandler
SysTick (15)	0X0000003C	SysTickIntHandler
GPIOPort-A (16)	0x00000040	GPIOIntHandler
Other programmable Interrupts (17 – 107	0x00000040 - 0x000001AC	IntDefaultHandler

As indicated in the table above:

- On reset, Stack Pointer will point to RAM location 0x20004900. RAM locations 0x20004900-0x20004000 are reserved for boot ROM.
- ResetIsr - is the function executed whenever M-Boot ROM is executed after reset.
- mbrom_nmi_interrupt_handler – is the function executed whenever there is an NMI during boot or as long as the NVIC base address points to 0x00000000 in M-Boot ROM.
- mbrom_hard_fault_isr_handler – is the function executed whenever there is a HARD FAULT condition detected by Cortext-M3 CPU during boot or as long as the NVIC base address points to 0x00000000 in M-Boot ROM.
  - **NOTE:** Memory Management Fault, Bus Fault and Usage Fault exceptions are disabled by default on reset in Cortex-M3 CPU and so is the case in M-Boot ROM. So if any of these errors occur during M-Boot ROM execution, they end up triggering a Hard Fault Exception.
- SysTickIntHandler – is the function used by Ethernet bootloaders for timing during EMAC boot.
- GpioIntHander – is the function used by the UART bootloader for AutoBaud calculation.

6.5.3 M-Boot ROM Version and Checksum Information

M-Boot ROM contains a version number located at 0x010001B0 occupying two bytes. This version is incremented each time M-Boot ROM code is modified. The next two bytes starting from 0x010001B2 contain the month and year (MM/YY in decimal) that the boot code was released. The next 0x84 bytes contain a checksum value for M-Boot ROM. A proprietary checksum algorithm is used on this device to allow boot ROM to be tested more efficiently. The checksum algorithm used is also embedded in ROM and will be published along with the device's boot ROM sources.

Table 6-3. M-Boot ROM Version and Checksum Information

Address	Contents
0x010001B0 – 0x010001B1	Revision No = 0x0100
0x010001B2 – 0x010001B3	Revision Date = 0x1012 (Month / Year) = Oct 2012
0x010001B4 – 0x01000238	Checksum

6.5.4 M-Boot ROM RAM Initialization

RAM memories on the master subsystem and control subsystem on Concerto devices must be zero-initialized before using the RAM for the first time to avoid any ECC and Parity errors. Refer to the *Internal Memory* chapter for more details on RAM ECC and Parity.

Copyright © 2012–2019, Texas Instruments Incorporated

During the device initialization process, M-Boot ROM reads the MRESC register and if the reset cause is POR, then all of the master subsystem RAMs are zero-initialized and for all the other reset causes, only M-Boot ROM stack memory is zero-initialized.

6.5.5 M-Boot ROM RAM Usage

M-Boot ROM uses part of C2 RAM, which is unsecured memory for stack, data and to log boot status. The first 0x900 bytes from start of C2 RAM start address (0x20004000) is reserved for boot ROM. Applications are free to reuse this memory after boot ROM execution is completed. However, that this part of C2 RAM will be zero-initialized by M-Boot ROM if it is rerun because of a reset which occurs while the application is running. In other words, if an application is using the first 0x900 bytes of C2 RAM, then it cannot expect the contents to be preserved between resets. This is extremely important for a debugger reset and software reset.

6.5.5.1 M-Boot ROM Stack

The C2 memory range 0x20004584 to 0x20004900 is allocated for the M-Boot ROM stack.

6.5.5.2 M-Boot ROM Data Section

The M-Boot ROM data section is located in C2 memory from address 0x20004004 to 0x20004583.

6.5.5.3 M-Boot ROM Boot Status

The first location in C2 RAM is reserved by M-Boot ROM boot status. This location is used for boot ROM to log status of different events that occur in the system during boot and applications can use this status to take necessary actions.

This location is preserved during master subsystem SW and debugger resets unlike the boot ROM data and stack memory range which is zero-initialized for these resets.

User applications can reuse this memory location after it reads the device boot status or it can leave this location reserved if there is a need to preserve boot status between resets.

Note that M-Boot ROM itself doesn't use this boot status for any purpose this is only provided to let applications know about it.

More details on the boot status is given in Section 6.5.12.

6.5.6 M-Boot ROM User OTP

The boot ROM user configurable DCSM OTP field descriptions and memory addresses are detailed in Table 6-4 and Table 6-5. Each field is provided with a description of its purpose and associated ECC information..

Table 6-4. REV0 – User Configurable DCSM OTP Fields

Address	Description	Notes
0x680800	OTP Security LOCK	ECC is shared for these fields, so they must be programmed at the same time.
0x680804	EMAC MAC ID 0	
0x680808	EMAC MAC ID 1	ECC is shared for these fields, so they must be programmed at the same time.
0x68080C	Reserved	

Table 6-5. REVA – User Configurable DCSM OTP Fields

Address	Description	Notes
0x680800	OTP Security LOCK	ECC is shared for these fields, so they must be programmed at the same time.
0x680804	Reserved	
0x680808	Reserved	ECC is shared for these fields, so they must be programmed at the same time.
0x68080C	Z2 FLASH ENTRY POINT	

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-5. REVA – User Configurable DCSM OTP Fields (continued)

Address	Description	Notes
0x680810	EMAC REG0 Address	ECC is shared for these fields, so they must be programmed at the same time.
0x680814	EMAC REG1 Address	
0x680818	Reserved	ECC is shared for these fields, so they must be programmed at the same time.
0x68081C	Main Oscillator Clock Frequency	
0x680820	Reserved	ECC is shared for these fields, so they must be programmed at the same time.
0x680824	Alternate Boot mode Pin Config	
0x680828	Reserved	ECC is shared for these fields, so they must be programmed at the same time.
0x68082C	OTP ENTRY POINT	

6.5.7 M-Boot ROM Entry Points

This section gives details about the entry point addresses for various boot modes supported by M-Boot ROM. These entry points tell M-Boot ROM where to branch to at the end of booting as per the boot mode selected. Table 6-6 lists and describes the M-Boot ROM entry point addresses.

Table 6-6. M-Boot ROM Entry Points

Name	Address	Size (x4)	Description
FLASH_ENTRY_POINT (in Z1)	0x200030	4	Default flash entry point for boot-to-flash option
CUSTOMER_OTP_Z2_FLASH_ENTRY_POINT	Configurable	4	Alternate flash entry point can be programmed in OTP location 0x68080C
RAM_ENTRY_POINT	0x20005000	4	Default RAM entry point for boot-to-RAM option
OTP_ENTRY_POINT	0x68082C	4	Default OTP entry point for boot-to-OTP option

6.5.7.1 M-Boot ROM: Boot-to-Flash, Flash Entry Point

The M-Boot ROM Flash entry point by default is fixed to Z1 (Zone 1) flash address 0x00200030. This location will be referred to as M_BOOT_ROM_Z1_FLASH_ENTRY_POINT further in this document. This means that if a user selects boot-to-flash option using the boot mode GPIO, then M-Boot ROM branches to location 0x00200030 in Z1 of master subsystem flash memory. However, there is an option for the user to change this default entry point location by programming location 0x0068100C in the customer OTP with an alternate entry point address.

For example: if a user wants to change the default flash entry point to a Z2 (Zone 2) flash address then the user can program this location with a valid Z2 flash address. This option allows user to force device to always boot to a Z2 flash address.

Notes:

- This location 0x0068080C is located in customer OTP and is one time programmable.
- If a user doesn't want to allow anyone to change the default flash entry point, then such a user can program 0x00000000 at location 0x0068080C. This value will make Boot ROM understand that user wants to use the default M_BOOT_ROM_Z1_FLASH_ENTRY_POINT only.
- If the contents of location 0x0068080C is equal to 0x00000000 or equal to 0xFFFFFFFF or fall outside the available master subsystem flash memory address range then M-Boot ROM defaults to M_BOOT_ROM_Z1_FLASH_ENTRY_POINT address.

6.5.7.2 M-Boot ROM: Boot-to-RAM, RAM EntryPoint

The M-Boot ROM Ram entry point by default is fixed to 0x20005000 in C2 RAM. This location will be referred to as M_BOOT_ROM_RAM_ENTRY_POINT further in this document.

This means if a user selects boot to RAM boot mode option using boot mode GPIO, then M-Boot ROM branches to location 0x20005000 in C2 RAM. User applications which use this option must have their main function located at this address or have a branch to main() instruction at this location.

Copyright © 2012–2019, Texas Instruments Incorporated

The boot to RAM option is mainly helpful during code development.

6.5.7.3 M-Boot ROM: Serial Boot Mode Entry Point

When using this boot mode option, there is a provision in the serial boot data transfer protocol for users to provide with the entry point. If no entry point is provided, then M-Boot ROM will branch to the same address where the user has asked M-Boot ROM to load the application in RAM.

Please refer to Section 6.5.15.1.4 for more details on serial boot mode data transfer protocol.

This entry point will be referred to as M_BOOT_ROM_SERIAL_BOOT_MODE_ENTRY_POINT further below in this document.

6.5.7.4 M-Boot ROM CAN Boot Mode Entry Point

When using this boot mode option, there is provision in the CAN boot data transfer protocol for users to provide with the entry point. If no entry point is provided by the user then M-Boot ROM will branch to the same address where user has asked it to load the application in RAM.

This entry point will be referred to as M_BOOT_ROM_CAN_BOOT_MODE_ENTRY_POINT further in this document.

Please refer to Section 6.5.15.3.3 for more details on CAN boot mode data transfer protocol.

6.5.7.5 M-Boot ROM EMAC Boot Mode Entry Point

When using this boot mode option there is no provision for users to mention an entry point or application load address because the EMAC bootloader on the device uses BOOTP and TFTP protocols to get the application data. When using this boot mode, M-Boot ROM by default, begins to load the application data from the peripheral on to the C2 RAM beginning from the M_BOOT_ROM_RAM_ENTRY_POINT address. At the end of data transfer, M-Boot ROM branches to the M_BOOT_ROM_RAM_ENTRY_POINT address to start the application downloaded from the EMAC peripheral.

Please refer to Section 6.5.15.2 for more details on EMAC bootload protocols.

6.5.7.6 M-Boot ROM Parallel Boot Mode Entry Point

When using this boot mode, there is a provision for the user to provide an entry point to tell M-Boot ROM where to branch to at the end of booting.

This entry point will be referred to as M_BOOT_ROM_PARALLEL_BOOT_MODE_ENTRY_POINT in this document.

Please refer to Section 6.5.15.4 for more details on PARALLEL boot load protocol.

6.5.7.7 M-Boot ROM SSI0 Master Boot Mode Entry Point

When using this boot mode, there is a provision for the user to provide an entry point to tell M-Boot ROM where to branch to at the end of booting. This entry point will be referred to as M_BOOT_FROM_SSI0_MASTER_MODE_ENTRY_POINT in this document. Please refer to Section 6.5.15.1.2 for more details on SSI0 boot load protocol.

6.5.7.8 M-Boot ROM I2C0 Master Boot Mode Entry Point

When using this boot mode, there is a provision for the user to provide an entry point to tell M-Boot ROM where to branch to at the end of booting. This entry point will be referred to as M_BOOT_FROM_I2C0_MASTER_MODE_ENTRY_POINT in this document. Please refer to Section 6.5.15.1.3 for more details on I2C0 boot load protocol.

Copyright © 2012–2019, Texas Instruments Incorporated

6.5.7.9 M-Boot ROM: Boot-to-OTP, OTP EntryPoint

The M-Boot ROM user OTP entry point by default is fixed to 0x68082C in OTP. This location is referred to as M_BOOT_ROM_OTP_ENTRY_POINT in this document. If a user selects boot-to-OTP boot-mode option using boot mode GPIO pins, then M-Boot ROM branches to location 0x68082C in OTP. User applications which use this option must have their main() function located at this address or have a branch to main() at this location. The boot-to-OTP option should be used if a custom boot loader is desired in the application.

6.5.8 M-Boot ROM Clock Initialization

On this device, PLL is disabled and bypassed by default on power-up or after an external reset. M-Boot ROM doesn't enable PLL and keeps it at its default state on power or after an external reset. M-Boot ROM, however, configures input clocks for both the master subsystem and control subsystem by modifying the SYSDIVSEL divider and M3SSCLK dividers as below. Please refer to the Clocking section of the *System Control and Interrupts* chapter for more details on these clock dividers.

Table 6-7. M-Boot ROM Clock Settings

Divider	Default on Power up or on External reset	M-Boot ROM setting
SYSDIVSEL	Divide by 8	Divide by 1
	(PLLSYSCLK = MainOscClock/8)	(PLLSYSCLK = MainOscClock/1)
M3SSDIVSEL	Divide by 4	Divide by 1
	(M3SSCLK = PLLSYSCLK/4)	(M3SSCLK = PLLSYSCLK = MainOscClk)

M-Boot ROM modifies the dividers as shown in Table 6-7 for all the reset types that pull external reset input to reset the master subsystem, except for a debugger reset or software reset. Refer to the Resets section in the *System Control and Interrupts* chapter for further discussion.

NOTE: Configuring the above dividers make a master subsystem and control subsystem RUN at the same frequency which is equal to MAINOSC frequency selected by user. So users should be aware of this fact while selecting a MAINOSC clock frequency. Remember that PLL is by passed during normal boot.

6.5.9 M-Boot ROM GPIO Assignments for Each Boot Mode

The table below gives information on the GPIOs used for each boot mode on M-Boot ROM. More details on the boot mode are provided further in this document.

Table 6-8. M-Boot ROM Boot Mode GPIO Assignments

M-Boot ROM Boot Mode	Peripheral	Boot Function Name for pin	Direction	GPIO(s) used	Pin Mux Assignment		
					Peripheral Mode	Alternate Mode	Core Select
Serial Boot Mode	UART0	UART0_RX	Input	PA0_GPIO0	1	0(default)	Master(default)
		UART0_TX	Ouput	PA1_GPIO1	1	0(default)	Master(default)
	I2C0	I2C0_CLK	Input	PB2_GPIO10	1	0(default)	Master(default)
		I2C_DATA	BI-Directional	PB3_GPIO11	1	0(default)	Master(default)
	SSI0	SSI0_CS	Input	PA3_GPIO3	1	0(default)	Master(default)
		SSI0_CLK	Input	PA2_GPIO2	1	0(default)	Master(default)
		SSI0_TX	Output	PA5_GPIO5	1	0(default)	Master(default)
		SSI0_RX	Input	PA4_GPIO4	1	0(default)	Master(default)
Parallel Boot Mode	GPIO (s)	D0	Input	PA0_GPIO0	0(default)	0(default)	Master(default)
		D1	Input	PA1_GPIO1	0(default)	0(default)	Master(default)
		D2	Input	PA2_GPIO2	0(default)	0(default)	Master(default)
		D3	Input	PA3_GPIO3	0(default)	0(default)	Master(default)
		D4	Input	PA4_GPIO4	0(default)	0(default)	Master(default)
		D5	Input	PA5_GPIO5	0(default)	0(default)	Master(default)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-8. M-Boot ROM Boot Mode GPIO Assignments (continued)

M-Boot ROM Boot Mode	Peripheral	Boot Function Name for pin	Direction	GPIO(s) used	Pin Mux Assignment		
		D6	Input	PB0_GPIO8	0(default)	0(default)	Master(default)
		D7	Input	PB1_GPIO9	0(default)	0(default)	Master(default)
		HOST_CTRL	Input	PE3_GPIO27	0(default)	0(default)	Master(default)
		DSP_CTRL	Ouput	PE2_GPIO26	0(default)	0(default)	Master(default)
CAN Boot Mode	CAN0	CAN0_RX	Input	PB4_GPIO12	5	0(default)	Master(default)
		CAN_TX	Output	PB5_GPIO13	5	0(default)	Master(default)
Ethernet Boot Mode	EMAC0	MII_TXD3	Output	PC4_GPIO68	3	0(default)	Master(default)
		MII_TXD2	Output	PH3_GPIO51	9	0(default)	Master(default)
		MII_TXD1	Output	PH4_GPIO52	9	0(default)	Master(default)
		MII_TXD0	Output	PH5_GPIO53	9	0(default)	Master(default)
		MII_RXD3	Input	PF5_GPIO37	3	0(default)	Master(default)
		MII_RXD2	Input	PG0_GPIO40	Alternate	12	Master(default)
		MII_RXD1	Input	PG1_GPIO41	Alternate	12	Master(default)
		MII_RXD0	Input	PH1_GPIO49	Alternate	12	Master(default)
		MII_TXER	Output	PG7_GPIO47	3	0(default)	Master(default)
		MII_RXDV	Input	PG3_GPIO43	Alternate	12	Master(default)
		MII_MDIO	Bi-directional	PE6_GPIO30	Alternate	12	Master(default)
		MII_TXEN	Output	PH6_GPIO54	Alternate	12	Master(default)
		MII_TXCK	Input	PH7_GPIO55	Alternate	12	Master(default)
		MII_RXER	Input	PJ0_GPIO56	3	0(default)	Master(default)
		MII_RXCK	Input	PJ2_GPIO58	Alternate	12	Master(default)
		MII_MDC	Output	PJ3_GPIO59	Alternate	12	Master(default)
		MII_COL	Input	PJ4_GPIO60	Alternate	12	Master(default)
		MII_CRS	Input	PJ5_GPIO61	Alternate	12	Master(default)
		MII_PHYINTRn	Input	PJ6_GPIO62	Alternate	12	Master(default)
		MII_PHYRSTn	Output	PJ7_GPIO63	Alternate	12	Master(default)
Boot from SSI0 master (Boot Mode 9	SSI0	SSI0_CS	Input	PA3_GPIO3	1	0(default)	Master(default)
		SSI0_CLK	Input	PA2_GPIO2	1	0(default)	Master(default)
		SSI0_TX	Output	PA5_GPIO5	1	0(default)	Master(default)
		SSI0_CRX	Input	PA4_GPIO4	1	0(default)	Master(default)
Boot from I2C0 master (Boot mode 10)	I2C0	I2C0_CLK	Input	PB2_GPIO10	1	0(default)	Master(default)
		I2C0_DATA	Bi-directional	PB3_GPIO11	1	0(default)	Master(default)

6.5.10 M-Boot ROM Functional Flow

As shown in Table 6-2, ResetISR is the main function that is called whenever M-Boot ROM is executed after a reset. This section explains the flow of M-Boot ROM following the function call sequence.

6.5.10.1 RESETISR()

- Call mbrom_change_dividers_checkfast_boot()
  - If the source of the reset is either POR / XRSn
    - Configure M3SSCLK and SYSDIVSEL = OSCCLK
    - Configure flash to power-up faster
- CSM initialization
  - Read OTPSECLOCK
  - Read Z1 CSM and Z2 CSM registers
- Device configuration initialization
- If the source of the reset is POR, initialize all RAMs (Cx RAM, SxRAM, M2CMSGRAM) to 0x0000
- If the source of the reset is NOT POR, zero-out stack used for boot ROM

Copyright © 2012–2019, Texas Instruments Incorporated

- Enable NMI
- Call mbrom_init_device()

6.5.10.2 MBROM_INIT_DEVICE()

- Call mbrom_master_system_init ()
  - Call mbrom_var_init()
  - If the source of the reset is POR, then clear the POR and the XRSn flag
  - If the source of the reset is XRSn, then clear the XRSn flag
  - If the source of the reset is a FUSE error, then enable watchdog0 and trigger watchdog reset
  - If the source of the reset is HWBIST, then branch to the OTP HWBIST reset handler
- Bring the control and analog subsystem out of reset
- If EMU0 =0 and EMU1 = 1, then put the device in wait in reset mode.
- Configure BOOT MODE pins and read device boot mode using configured BOOT MODE pins (or) using default boot mode pins
- If boot mode = boot-to-flash, branch to the flash start address 0x200030 (or) branch to the alternate flash start address stored in CUSTOMER_OTP_Z2_FLASH_ENTRY_POINT (0x68080C)
- If boot mode = serial boot (I2C0, SSI0 and UART0) mode, then execute the following functions:
  - ConfigureDevice()
  - PickInterface()
  - Updater()
  - Download application data from any of the serial peripherals (UART0 / SSI0 / I2C0) onto RAM
  - Call mbrom_start_app M_BOOT_ROM_SERIAL_BOOT_MODE_ENTRY_POINT)
- If boot mode = boot to RAM mode, then call mbrom_start_app(RAM_ENTRY_POINT)
- If boot mode = boot to EMAC mode, then
  - Download application data from the EMAC peripheral to RAM
  - Call mbrom_start_app(M_BOOT_ROM_EMAC_BOOT_MODE_ENTRY_POINT)
- If boot mode = boot to CAN mode, then
  - Download application data from the CAN peripheral to RAM
  - Call mbrom_start_app(M_BOOT_ROM_CAN_BOOT_MODE_ENTRY_POINT)
- If boot mode = boot to parallel mode, then
  - Download application data using parallel GPIOs to RAM
  - Call mbrom_start_app(M_BOOT_ROM_PARALLEL_BOOT_MODE_ENTRY_POINT)
- If boot mode = boot to SSI0 master mode, then
  - Download application data using the SSI0 master mode to RAM
  - Call mbrom_start_app(M_BOOT_ROM_SSI0_MASTER_BOOT_MODE_ENTRY_POINT)
- If boot mode = boot to I2C master mode, then
  - Download application data using the I2C master mode to RAM
  - Call mbrom_start_app(M_BOOT_ROM_I2C0_MASTER_BOOT_MODE_ENTRY_POINT)
- If boot mode is boot_to_OTP
  - Call mbrom_start_app(M_BOOT_ROM_OTP_ENTRY_POINT
- If boot mode is invalid, then default to the boot-to-flash mode.

6.5.10.3 MBROM_START_APP(ENTRY_POINT_ADDRESS)

- Reset device configuration set up by M-Boot ROM for boot purposes.
  - Disable clock to all peripherals

Copyright © 2012–2019, Texas Instruments Incorporated

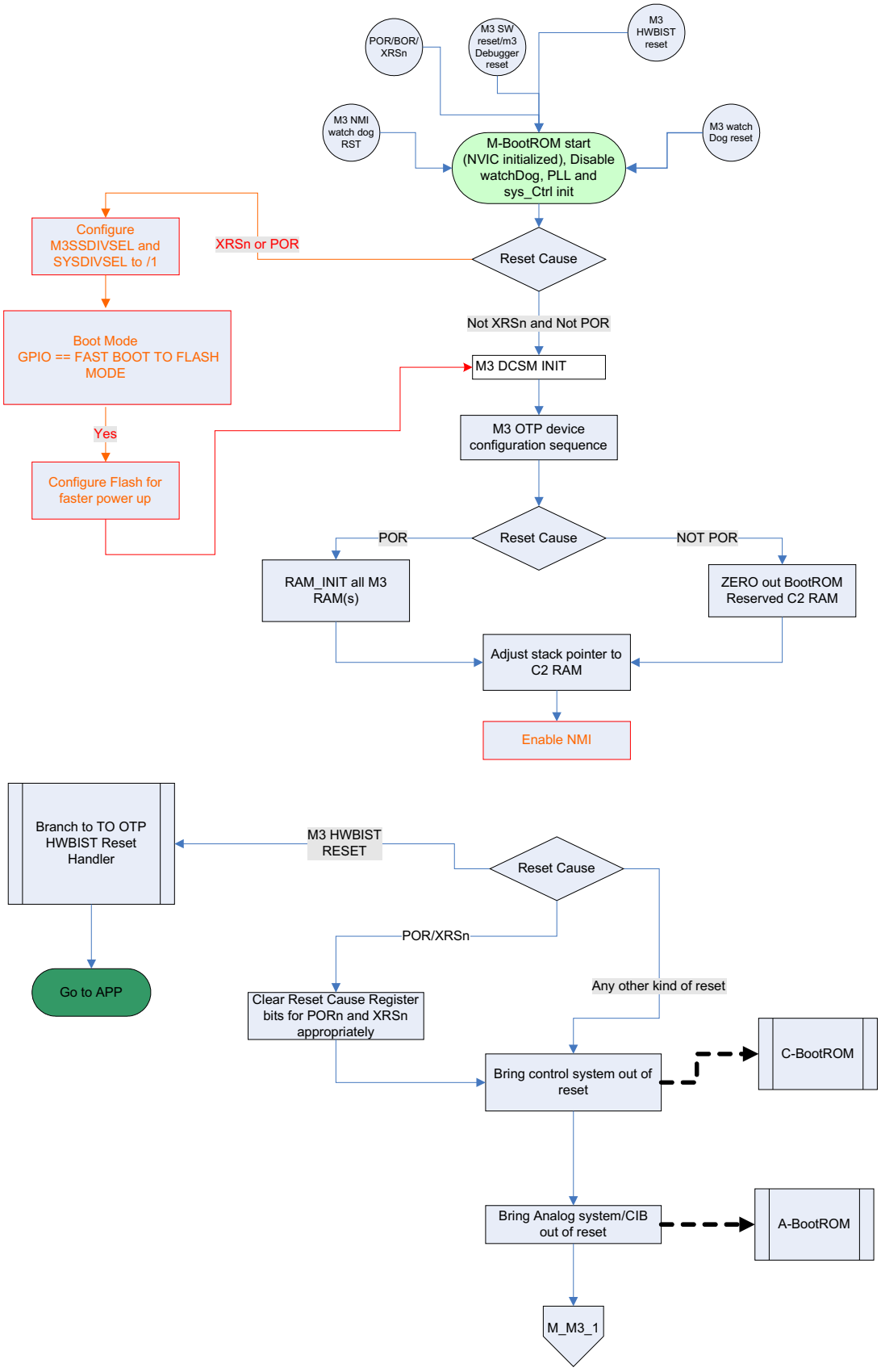
  - Reset all peripherals
  - Disable SysTick interrupt
  - Clear any pending interrupts
  - Reset NVIC Base address
  - Clear any pending faults
- Branch to the entry point address

6.5.11 M-Boot ROM Flow Diagram

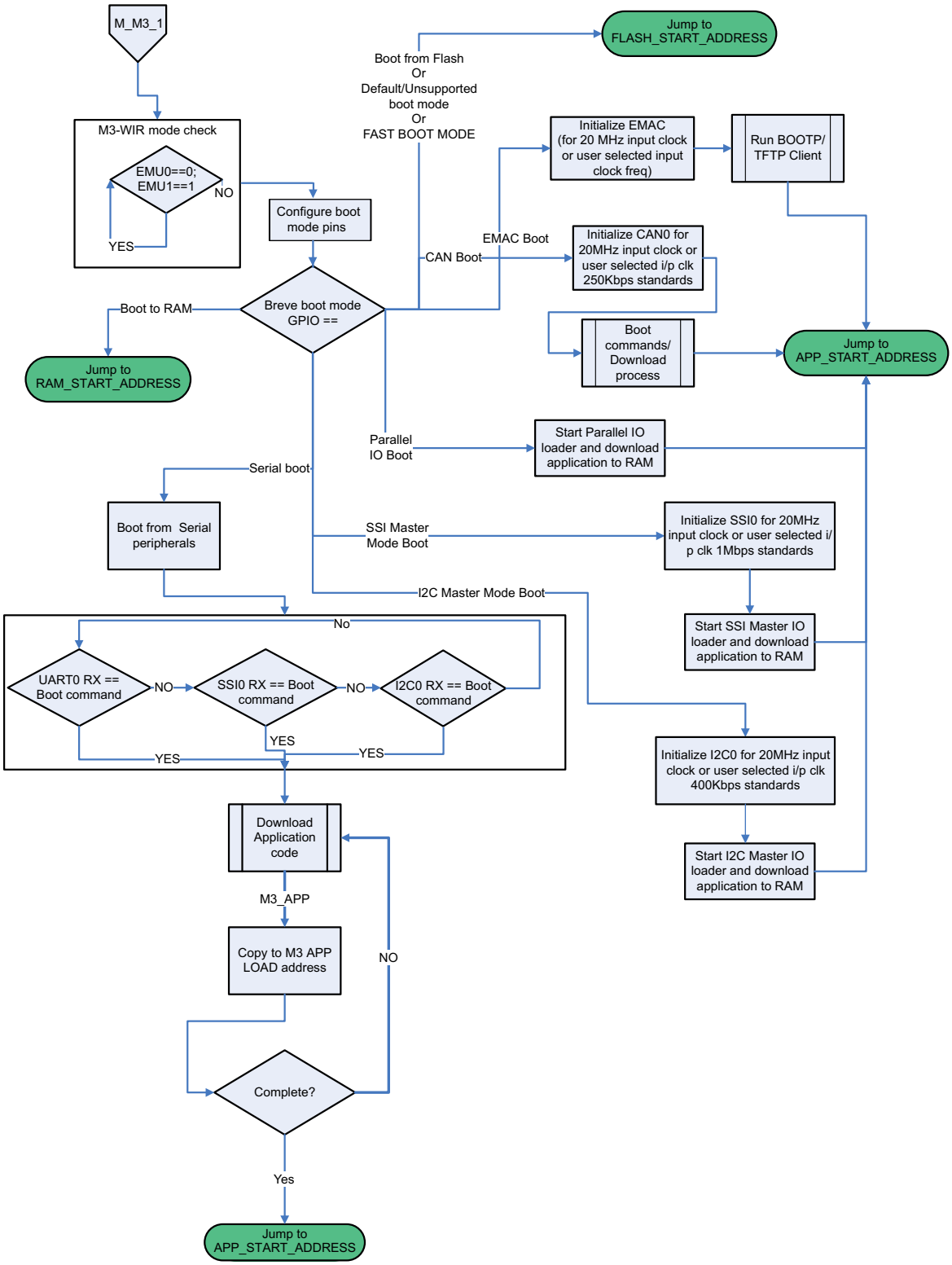
The M-Boot ROM flow diagram is shown below.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-3. M-Boot ROM Flow Diagram



Copyright © 2012–2019, Texas Instruments Incorporated



NOTE: Boot-to-OTP option is not shown in this flow chart, but it is similar to boot-to-RAM option. The boot-ROM upon decoding the boot-to-OTP option from the boot-mode GPIO pins jumps to the M_BOOT_ROM_OTP_ENTRY_POINT.

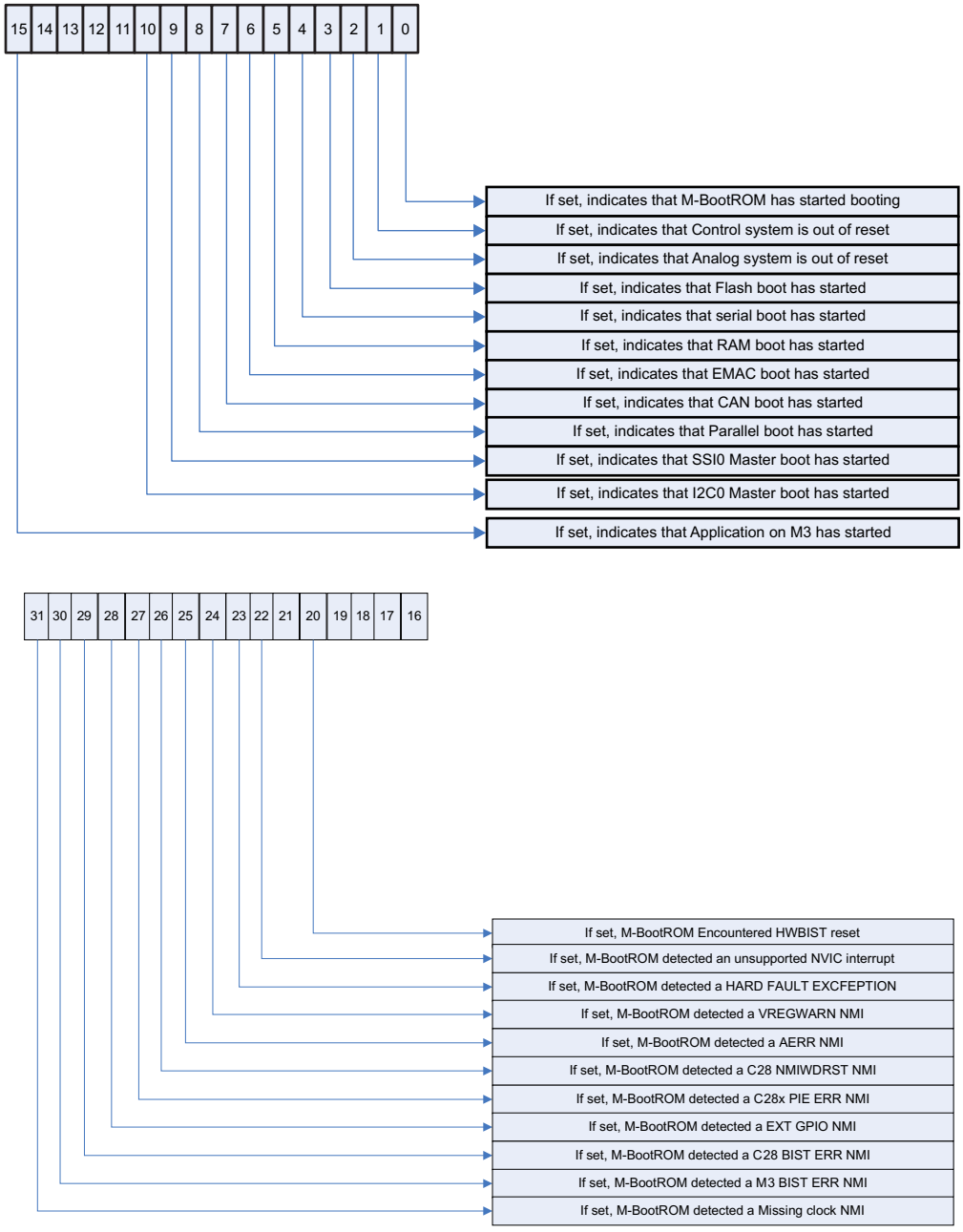
Copyright © 2012–2019, Texas Instruments Incorporated

6.5.12 M-Boot ROM Boot Status in RAM for Applications

M-Boot ROM keeps a record of different events that can occur during boot ROM execution, for applications to learn about the boot status. This is necessary because NMI and other exceptions are enabled by default in the device and will be explained further in this document. M-Boot ROM makes an effort to handle all the possible exceptions and continues to boot and start application properly. On start-up, the application can look at this boot status and take necessary actions per application's need and ability to handle these events.

As explained earlier, M-Boot ROM logs the boot status at the first location in C2 RAM, which is 0x20004000. The diagrams below give details on what it means when each bit is set in location 0x20004000. All the bits that don't have any status linked with them MUST be ignored.

Figure 6-4. M-Boot ROM Boot Status



Copyright © 2012–2019, Texas Instruments Incorporated

6.5.13 M-Boot ROM Reset Cause Handling

As shown in Figure 6-4, M-Boot ROM will be executed for any reset cause that resets the master subsystem CPU core. Table 6-9 gives details on how M-Boot ROM handles each reset.

Table 6-9. M-Boot ROM Reset Cause Handling

Reset Source	M-Boot ROM action
POR	ModeZero-initialize all master subsystem memories, CLOCK_INIT(s) and follow normal boot-up procedure
$\overline{\text{XRS}}$ Input	Zero-initialize M-Boot ROM stack memory, CLOCK_INIT(s) and follow normal boot-up procedure
M3 WDT0 reset	Zero-initialize M-Boot ROM stack memory, CLOCK_INIT(s) and follow normal boot-up procedure
M3 WDT1 reset	Zero-initialize M-Boot ROM stack memory, CLOCK_INIT(s) and follow normal boot-up procedure
M3 NMIWD reset (M3 $\overline{\text{NMIRS}}$)	Zero-initialize M-Boot ROM stack memory, CLOCK_INIT(s) and follow normal boot-up procedure
M3 Software reset / debugger reset	Zero-initialize M-Boot ROM stack memory and follow normal boot-up procedure. Note that clock settings are not modified here.

CLOCK_INIT(s) in the above table means SYSDIVSEL and M3SSDIVSEL dividers are configured for divide by 1 operation, as explained in Section 6.5.8 above.

6.5.14 M-Boot ROM Exceptions Handling

Table 6-10 shows how M-Boot ROM handles different possible exceptions that can occur during boot.

Table 6-10. M-Boot ROM Exceptions Handling

Exception Source	Description	M-Boot ROM Action
CLOCKFAIL	CLKFAIL condition detected	Clear MNMI Flags and log the error in boot status location and continue to boot. MCLKSTS.MCLKFLG is not cleared here.
M3BISTERR	BIST error on M3	Clear NMI Flags and log the error in boot status location and continue to boot.
C28BISTERR	BIST error on C28X	Clear NMI Flags and log the error in boot status location and continue to boot.
EXTGPIO	External GPIO NMI	Clear MNMI Flags and log the error in boot status location and continue to boot.
C28PIENMIERR	NMI vector fetch mismatch on the control subsystem	Clear MNMI Flags and log the error in boot status location and continue to boot.
C28NMIWDRST	The control subsystem is reset by CNMIWD timer	Clear MNMI Flags and log the error in boot status location and continue to boot.
ACIBERR	ACIB access error	Clear MNMI Flags and log the error in boot status location and continue to boot.
Bus Fault/ Memory Management Fault/ Usage Fault	Triggers HARD FAULT exception to NVIC	Log the error in boot status and configure WDT0 and loop and let device reset on WDT0 timeout
Hard Fault	Triggers HARD FAULT exception to NVIC	Log the error in boot status and configure WDT0 and loop and let device reset on WDT0 timeout
Spurious/un supported NVIC interrupt	Triggered because of any errors	Log the error in boot status and configure WDT0 and loop and let device reset on WDT0 timeout.

Copyright © 2012–2019, Texas Instruments Incorporated

All the NMI exceptions in M-Boot ROM are handled as said above by the handler . **mbrom_nmi_interrupt_handler().**

The HardFault Exception in M-Boot ROM is handled as said above by the handler . **mbrom_hard_fault_isr_handler().**

Spurious NVIC interrupt in M-Boot ROM is handled as said above by the handler IntDefaultHandler().

6.5.15 M-Boot ROM Boot Modes

This section gives details on how M-Boot ROM configures the device for different supported boot modes and how M-Boot ROM handles data transfer in each boot mode. Serial Boot, EMAC Boot and CAN Boot mode data transfer protocols between Concerto and the Host device that sends boot data to M-Boot ROM are compatible to respective bootloaders in Stellaris devices.

The M-Boot ROM Parallel Boot data transfer protocol is compatible to the C2000 Parallel bootloader, found on Piccolo devices. This allows users to re-use existing LM Flash programmer HOST software collateral and other C2000 collateral to send boot data to Concerto devices.

6.5.15.1 M-Boot ROM Serial Boot Mode

There are three serial interfaces that can be used for communicating with the bootloader: UART0, SSI0, and I2C0. All three share a common protocol and differ only in the physical connections and signaling used to transfer the bytes of the protocol.

When serial boot mode is selected, M-Boot ROM polls each of these interfaces sequentially for the boot mode commands and boots from the interface on which it receives data.

6.5.15.1.1 UART Interface

The UART0 interface uses below pins to communicate with the bootloader.

UART0_RX	PA0_GPIO0, peripheral MODE -1
UART0_TX	PA1_GPIO1, peripheral MODE -1

The device communicating with the bootloader is responsible for driving the UART0_RX pin on the Concerto Microcontroller while the Concerto Microcontroller drives the UART0_TX pin.

The serial data format is fixed at 8 data bits, no parity, and one stop bit. An auto-baud feature is used to determine the baud rate at which data is transmitted. Note that the system clock must be at least 32 times the baud rate, this determines the maximum baud rate that can be used.

Max Baud Rate that can be used = M3SSCLK frequency/32
In M-Boot ROM M3SSCLK = PLLSYSCLK = MAINOSC Clock
So, max baud rate on this device = M AINOSC/32

When an application calls back to the ROM-based bootloader to start an update over the UART port, the auto-baud feature is bypassed, along with UART configuration and pin configuration. Therefore, the UART must be configured and the UART pins switched to their hardware function before calling the bootloader.

6.5.15.1.2 SSI INTERFACE

The SSI0 interface bootloader uses below GPIO pins for SSI communications.

SSI0_CS	PA3_GPIO3, peripheral MODE -1
SSI0_CLK	PA2_GPIO2, peripheral MODE -1
SSI0_TX	PA5_GPIO5, peripheral MODE -1
SSI0_RX	PA4_GPIO4, peripheral MODE -1

Copyright © 2012–2019, Texas Instruments Incorporated

The device communicating with the bootloader is responsible for driving the SSI0_RX, SSI0_CLK, and SSI0_CS pins, while the Concerto microcontroller drives the SSI0_TX pin.

The serial data format is fixed to the Motorola format with SPH set to 1 and SPO set to 1 (see the applicable Concerto family data sheet for more information on this format).

When an application calls back to the ROM-based bootloader to start an update over the SSI port, the SSI configuration and pin configuration is bypassed. Therefore, the SSI port must be configured and the SSI pins switched to their hardware function before calling the bootloader.

6.5.15.1.3 I2C Interface

The I2C0 interface bootloader uses the below GPIO pins for I2C communications.

I2C0_CLK	PB2_GPIO10, peripheral MODE -1
I2C0_DATA	PB3_GPIO11, peripheral MODE -1

The device communicating with the bootloader must operate as the I2C master and provide the I2C0_CLK signal. The I2C0_DATA pin is open-drain and can be driven by either the master or the slave I2C device. The I2C interface can run at up to 400 KHz, the maximum rate supported by the I2C protocol. The bootloader uses an I2C slave address of 0x42.

When an application calls back to the ROM-based bootloader to start an update over the I2C port, the I2C configuration and pin configuration is bypassed. Therefore, the I2C port must be configured, the I2C slave address set, and the I2C pins switched to their hardware function before calling the bootloader. Additionally, the I2C master must be enabled since it is used to detect start and stop conditions on the I2C bus.

6.5.15.1.4 M-Boot ROM Serial Boot Protocol

The bootloader uses well-defined packets on the serial interfaces to ensure reliable communications with the update program. The packets are always acknowledged or not acknowledged by the communicating devices. The packets use the same format for receiving and sending packets. This includes the method used to acknowledge successful or unsuccessful reception of a packet. While the actual signaling on the serial ports is different, the packet format remains independent of the method of transporting the data.

The following steps must be performed to successfully send a packet:

1. Send the size of the packet that will be sent to the device. The size is always the number of bytes of data + 2 bytes.
2. Send the checksum of the data buffer to help ensure proper transmission of the command. The checksum is simply a sum of the data bytes.
3. Send the actual data bytes.
4. Wait for a single-byte acknowledgment from the device that it either properly received the data or that it detected an error in the transmission.

The following steps must be performed to successfully receive a packet:

1. Wait for non-zero data to be returned from the device. This is important as the device may send zero bytes between a sent and received data packet. The first non-zero byte received will be the size of the packet that is being received.
2. Read the next byte which will be the checksum for the packet.
3. Read the data bytes from the device. There will be packet size - 2 bytes of data sent during the data phase. For example, if the packet size was 3, then there is only 1 byte of data to be received.
4. Calculate the checksum of the data bytes and ensure that it matches the checksum received in the packet.
5. Send an acknowledge (ACK) or not-acknowledge (NAK) to the device to indicate the successful or unsuccessful reception of the packet. An acknowledge packet is sent whenever a packet is successfully received and verified by the bootloader. A not-acknowledge packet is sent whenever a sent packet is detected to have an error, usually as a result of a checksum error or just malformed data in the packet. This allows the sender to re-transmit the previous packet.

Copyright © 2012–2019, Texas Instruments Incorporated

The following commands are used by the custom protocol:

Table 6-11. M-Boot ROM Serial Boot Commands

Serial No.	Command Name	Command Value	Description
1.	COMMAND_PING	0x20	This command is used to receive an acknowledge from the bootloader indicating that communication has been established. This command is a single byte. The format of the command is as follows: unsigned char ucCommand[1]; ucCommand[0] = COMMAND_PING;

2.	COMMAND_DOWNLOAD	0x21	This command is sent to the bootloader to indicate where to store data and how many bytes will be sent by the COMMAND_SEND_DATA commands that follow. The command consists of two 32-bit values that are both transferred MSB first. The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data that will be sent. This command should be followed by a COMMAND_GET_STATUS to ensure that the program address and program size were valid for the microcontroller running the bootloader.
			The format of the command is as follows:
			unsigned char ucCommand[9];
			ucCommand[0] = COMMAND_DOWNLOAD;
			ucCommand[1] = Program Address [31:24];
			ucCommand[2] = Program Address [23:16];
			ucCommand[3] = Program Address [15:8];
			ucCommand[4] = Program Address [7:0];
			ucCommand[5] = Program Size [31:24];
			ucCommand[6] = Program Size [23:16];
			ucCommand[7] = Program Size [15:8];
			ucCommand[8] = Program Size [7:0];

3.	COMMAND_RUN	0x22	This command is sent to the bootloader to transfer execution control to the specified address. The command is followed by a 32-bit value, transferred MSB first, that is the address to which execution control is transferred.
			The format of the command is as follows:
			unsigned char ucCommand[5];
			ucCommand[0] = COMMAND_RUN;
			ucCommand[1] = Run Address [31:24];
			ucCommand[2] = Run Address [23:16];
			ucCommand[3] = Run Address [15:8];
			ucCommand[4] = Run Address [7:0];

4.	COMMAND_GET_STATUS	0x23	This command returns the status of the last command that was issued. Typically, this command should be received after every command is sent to ensure that the previous command was successful or, if unsuccessful, to properly respond to a failure. The command requires one byte in the data of the packet and the bootloader should respond by sending a packet with one byte of data that contains the current status code The format of the command is as follows: unsigned char ucCommand[1]; ucCommand[0] = COMMAND_GET_STATUS; The following are the definitions for the possible status values that can be returned from the bootloader when COMMAND_GET_STATUS is sent to the microcontroller. COMMAND_RET_SUCCESS (= 0x40) COMMAND_RET_UNKNOWN_CMD (= 0x41)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-11. M-Boot ROM Serial Boot Commands (continued)

Serial No.	Command Name	Command Value	Description
			COMMAND_RET_INVALID_CMD (= 0x42) COMMAND_RET_INVALID_ADD (= 0x43)

5.	COMMAND_SEND_DATA	0x24	This command should only follow a COMMAND_DOWNLOAD command or another COMMAND_SEND_DATA command, if more data is needed. Consecutive send data commands automatically increment the address and continue programming from the previous location. The transfer size is limited by the maximum size of a packet, which allows up to 252 data bytes to be transferred at a time. The command terminates programming once the number of bytes indicated by the COMMAND_DOWNLOAD command has been received. Each time this function is called, it should be followed by a COMMAND_GET_STATUS command to ensure that the data was successfully programmed. If the bootloader sends a NAK to this command, the bootloader will not increment the current address which allows for retransmission of the previous data.
			The format of the command is as follows:
			unsigned char ucCommand[9];
			ucCommand[0] = COMMAND_SEND_DATA
			ucCommand[1] = Data[0];
			ucCommand[2] = Data[1];
			ucCommand[3] = Data[2];
			ucCommand[4] = Data[3];
			ucCommand[5] = Data[4];
			ucCommand[6] = Data[5];
			ucCommand[7] = Data[6];
			ucCommand[8] = Data[7];

6.	COMMAND_RESET	0x25	This command is used to tell the bootloader to start the application. The start address of application will be same as the download address given by user when using the COMMAND_DOWNLOAD command. This command can be used by user to start the downloaded application if the RUN address of the downloaded application is same as the DOWNLOADED address. If M-Boot ROM detects that there is no valid address for it to start the application then it will return COMMAND_RET_INVALID_ADR. If the RUN address is different from DOWNLOADED address then it is advised that user use the COMMAND_RUN command. The format of the command is as follows: unsigned char ucCommand[1]; ucCommand[0] = COMMAND_RESET;
Note: The ACK and NAK byte packets are defined below: ACK packet: byte0 will be 0x0 and byte1 will be 0xCC NAK packet: byte0 will be 0x0 and byte1 will be 0x33			

6.5.15.1.5 M-Boot ROM Serial Boot Load Functional Overview

The below function call sequence gives details of the flow when serial boot mode is selected on the device.

- ResetIsr()
- mbrom_init_device()
- mbrom_master_system_init ()
- mbrom_control_system_init()

Copyright © 2012–2019, Texas Instruments Incorporated

- mbrom_analog_system_init()
- M-Boot ROM WIR Mode Check – Please refer to the WIR Mode section in the *System Control and Interrupts* chapter.
- mbrom_get_bootmode()
  - If boot mode is boot_from_serial, then execute the following functions
    - ConfigureDevice()
    - PickInterface();
    - Update()
    - Download application data from any of the available serial communication peripherals (UART0/SSI0/I2C0) on to RAM.
- mbrom_start_app(M_BOOT_ROM_SERIAL_BOOT_MODE_ENTRY_POINT)

When performing an update via a serial port (UART0, SSI0, or I2C0), ConfigureDevice() is used to configure the serial ports (all the three interfaces) making them ready to be used to download application code on to device RAM.

Then PickInterface() function is called which will continuously poll UART0, SSI0 and I2C0 interfaces for a successful baud rate synchronization on UART0 or successful data reception on SSI0 or successful data reception on I2C0. If baud rate is successfully synchronized on UART0 using auto baud feature then UART0 interface is selected for boot. Otherwise either I2C0 or SSS0 interface is selected based on which interface received data. If UART0 auto baud is not successful and neither I2C0 and SSI0 interfaces received data then M-Boot ROM keeps waiting for one of the above events to occur

Once an interface is picked to boot, Update() function is called which sits in an endless loop accepting commands and updating the application code when requested. All transmissions from this main routine use the packet handler functions (SendPacket(), ReceivePacket(), AckPacket(), and NakPacket()). Once the update is complete, the application can be started by issuing a COMMAND_RUN command or COMMAND_RESET command.

6.5.15.1.5.1 Packet Handling

The bootloader uses the SendPacket() function in order to send a packet of data to another device. This function encapsulates all of the steps necessary to send a valid packet to another device including waiting for the acknowledge or not-acknowledge from the other device.

Received packets use the same format as sent packets. The bootloader uses the ReceivePacket() function in order to receive or wait for a packet from another device. This function does not take care of acknowledging or not-acknowledging the packet to the other device. This allows the contents of the packet to be checked before sending back a response.

The steps performed while sending a packet are as listed in Section 6.5.15.1.4.

The steps necessary to acknowledge reception of a packet are implemented in the AckPacket() function. Acknowledge bytes are sent out whenever a packet is successfully received and verified by the bootloader.

A not-acknowledge byte is sent out whenever a sent packet is detected to have an error, usually as a result of a checksum error or just malformed data in the packet. This allows the sender to re-transmit the previous packet.

6.5.15.1.5.2 Transport Layer

The bootloader supports updating via the I2C0, SSI0, and UART0 ports which are available on Concerto microcontrollers. The SSI port has the advantage of supporting higher and more flexible data rates but it also requires more connections to the microcontroller. The UART has the disadvantage of having slightly lower and possibly less flexible rates. However, the UART requires fewer pins and can be easily implemented with any standard UART connection. The I2C interface also provides a standard interface, only uses two wires, and can operate at comparable speeds to the UART and SSI interfaces.

Copyright © 2012–2019, Texas Instruments Incorporated

The I2C handling functions are I2CSend(), I2CReceive(), and I2CFlush() functions. The device communicating with the bootloader must operate as the I2C master and provide the I2C0_CLOCK signal. The I2C0_DATA pin is open drain and can be driven by either the master or the slave I2C device.

6.5.15.1.5.2.1 I2C Transport

The bootloader supports updating via the I2C0, SSI0, and UART0 ports which are available on Concerto microcontrollers. The SSI port has the advantage of supporting higher and more flexible data rates but it also requires more connections to the microcontroller. The UART has the disadvantage of having slightly lower and possibly less flexible rates. However, the UART requires fewer pins and can be easily implemented with any standard UART connection. The I2C interface also provides a standard interface, only uses two wires, and can operate at comparable speeds to the UART and SSI interfaces.

The I2C handling functions are I2CSend(), I2CReceive(), and I2CFlush() functions. The device communicating with the bootloader must operate as the I2C master and provide the I2C0_CLOCK signal. The I2C0_DATA pin is open drain and can be driven by either the master or the slave I2C device.

6.5.15.1.5.2.2 SSI Transport

The SSI handling functions are SSISend(), SSIReceive(), and SSIFlush(). The device communicating with the bootloader is responsible for driving the SSI0_RX,SSI0_CLK, and SSI0_CS pins, while the Concerto microcontroller drives the SSI0_TX pin. The format used for SSI communications is the Motorola format with SPH set to 1 and SPO set to 1 (see Stellaris Family data sheet for more information on this format). The SSI interface has a hardware requirement that limits the maximum rate of the SSI clock to be at most 1/12 the frequency of the microcontroller running the bootloader.

6.5.15.1.5.2.3 UART Transport

The UART handling functions are UARTSend(), UARTReceive(), and UARTFlush().The device communicating with the bootloader is responsible for driving the UART0_RX pin on the Concerto microcontroller, while the Concerto microcontroller drives the UART0_TX pin.

While the baud rate is flexible, the UART serial format is fixed at 8 data bits, no parity, and one stop bit. The baud rate used for communication can either be auto-detected by the bootloader, if the auto-baud feature is enabled, or it can be fixed at a baud rate supported by the device communicating with the bootloader. The only requirement on baud rate is that the baud rate should be no more than 1/32 the frequency of the microcontroller that is running the bootloader. This is the hardware requirement for the maximum baud rate for a UART on any Concerto microcontroller.

When using a fixed baud rate, the frequency of the crystal connected to the microcontroller must be specified. Otherwise, the bootloader will not be able to configure the UART to operate at the requested baud rate.

The bootloader provides a method to automatically detect the baud rate being used to communicate with it. This automatic baud rate detection is implemented in the UARTAutoBaud() function. The auto-baud function attempts to synchronize with the updater application and indicates if it is successful in detecting the baud rate or if it failed to properly detect the baud rate. The PickInterface() function makes multiple calls to UARTAutoBaud() to attempt to retry the synchronization if the first call fails. In M-Boot ROM, the bootloader will wait forever for a valid synchronization pattern from the host.

6.5.15.2 M-Boot ROM EMAC Boot Mode

This section gives details on EMAC boot mode in M-Boot ROM on the Concerto devices.

When using the Ethernet interface to communicate with the bootloader, the BOOTP and TFTP protocols are utilized. By using standard protocols, the bootloader will co-exist in a normal Ethernet environment without causing any problems (other than using a small amount of network bandwidth).

Copyright © 2012–2019, Texas Instruments Incorporated

The bootstrap protocol (BOOTP), a predecessor to the DHCP protocol, is used to discover the IP address of the client, the IP address of the server, and the name of the firmware image to use. BOOTP uses UDP/IP packets to communicate between the client and the server; the bootloader acts as the client. First, it will send a BOOTP request using a broadcast message. When the server receives the request, it will reply, thereby informing the client of its IP address, the IP address of the server, and the name of the firmware image. Once this reply is received, the BOOTP protocol has completed.

Then, the trivial file transfer protocol (TFTP) is used to transfer the firmware image from the server to the client. TFTP also uses UDP/IP packets to communicate between the client and the server, and the bootloader also acts as the client in this protocol. As each data block is received, it is programmed into flash. Once all data blocks are received and programmed, the device will automatically start running the new firmware image.

Note: When using the Ethernet update, the bootloader can only program images to the RAM beginning from 0x20005000 since there is no mechanism in BOOTP to specify the address to program the image.

The following IETF specifications define the protocols used by the Ethernet update mechanism:

RFC951 (http://tools.ietf.org/html/rfc951.html) defines the bootstrap protocol

RFC1350 (http://tools.ietf.org/html/rfc1350.html) defines the trivial file transfer protocol.

6.5.15.2.1 M-Boot ROM EMAC Interface IO

```
MII_TXD3          -     PC4_GPIO68, peripheral Mode - 3
MII_MDIO          -     PE6_GPIO30, Peripheral Mode - Alternate Mode 12
MII_RXD3          -     PF5_GPIO37, peripheral Mode - 3
MIIRXD2           -     PG0_GPIO40, Peripheral Mode - Alternate Mode 12
MIIRXD1           -     PG1_GPIO41, Peripheral Mode - Alternate Mode 12
MII_RXDV          -     PG3_GPIO43, Peripheral Mode - Alternate Mode 12
MII_TXER          -     PG7_GPIO47, Peripheral Mode - 3
MIIRXD0           -     PH1_GPIO49, peripheral Mode - Alternate mode 12
MII_TXD2          -     PH3_GPIO51, Peripheral Mode - 9
MII_TXD1          -     PH4_GPIO52, Peripheral Mode - 9
MII_TXD0          -     PH5_GPIO53, Peripheral Mode - 9
MIITXEN           -     PH6_GPIO54, peripheral Mode - Alternate mode 12
MIITXCK           -     PH7_GPIO55, peripheral Mode - Alternate mode 12
MII_RXER          -     PJ0_GPIO56, Peripheral Mode - 3
MII_RXCK          -     PJ2_GPIO58, Peripheral Mode - Alternate Mode 12
MIIMDC            -     PJ3_GPIO59, peripheral Mode - Alternate mode 12
MIICOL            -     PJ4_GPIO60, peripheral Mode - Alternate mode 12
MII_CRS           -     PJ5_GPIO61, peripheral Mode - Alternate mode 12
MII_PHYINTRn      -     PJ6_GPIO62, peripheral Mode - Alternate mode 12
MIIPHYRSTn        -     PJ7_GPIO63, Peripheral Mode - Alternate Mode 12
```

Note: EMAC interface is configured to operate during boot with the assumption that MAIN OSC clock frequency is 20 MHz.

6.5.15.2.2 M-Boot ROM EMAC ID Configuration

M-Boot ROM follows below procedure to determine the EMAC ID of the device, when using EMAC boot mode.

The locations below in Customer OTP memory must be programmed by user to let M-Boot ROM use the EMAC ID provided by user.

Locations to be programmed will be referred to as below and point to the addresses as shown below:

CUSTOMER_OTP_EMAC_REG0_ADDR = 0x680810
CUSTOMER_OTP_EMAC_REG1_ADDR = 0x680814

A MAC address of 12:34:56:78:9A:BC should get stored/programmed as:
CUSTOMER_OTP_EMAC_REG0_ADDR = 0x00563412
CUSTOMER_OTP_EMAC_REG1_ADDR = 0x00BC9A78

Copyright © 2012–2019, Texas Instruments Incorporated

The top byte of each EMACID location is all 0x00. Only the lower 3 bytes of each EMACID location are used as shown above. If the top byte of either EMACID location is not '0', then the default EMAC ID will be used by M-Boot ROM.

Default EMAC ID used by M-Boot ROM – 0x1A:B6:00:64:00

6.5.15.2.3 M-Boot ROM EMAC Boot Functional Flow

When performing an Ethernet bootloader, EnetConfig()is used to configure the Ethernet controller, making it ready to be used to download the application onto the master subsystem RAM. Then, Update() function calls into UpdateBOOTP() function which begins the process of the application download.

The below function call sequence gives details of the flow when EMAC boot mode is selected on the device.

- ResetIsr()
- mbrom_init_device()
- mbrom_master_system_init ()
- mbrom_control_system_init()
- mbrom_analog_system_init()
- M-Boot ROM WIR Mode Check – Please refer to WIR Mode section in the *System Control and Interrupts* Chapter.
- mbrom_get_bootmode()
  - EnetConfig();
  - SelectEnet();
  - Update();
- mbrom_start_app(M_BOOT_ROM_RAM_ENTRY_POINT)

6.5.15.3 M-Boot ROM CAN Boot Mode

This section gives details on the CAN boot mode in M-Boot ROM on the Concerto devices.

When performing a CAN update the bootloader calls ConfigureCAN() to configure the CAN controller and prepare the bootloader to download the application on to device RAM. The CAN update mechanism allows the bootloader to be entered from a functioning CAN application as well from startup when no application has been downloaded to the microcontroller. The bootloader provides the main routine for performing the CAN update in the UpdaterCAN() function which is used in both cases.

When the device enters the bootloader from a running CAN application, the bootloader will not reconfigure the CAN clocks or bit timing and will assume that they have been configured as expected by the application running on device. The bootloader assumes that the application has taken the device off of the CAN network by putting it in "Init mode" but left the CAN bit timings untouched.

When the bootloader is run without an application, which is from M-Boot ROM, it is necessary to configure the CAN bit rate using the default CAN clocking which is given below.

6.5.15.3.1 CAN Bus Clocking

M-Boot ROM configures the CAN interface for 250Kbps baud rate assuming that the input clock frequency to CAN is at 20 MHz..

Input clock frequency to CAN during boot is equal to M3SSCLK which is equal to MAINOSC frequency.

6.5.15.3.2 M-Boot ROM CAN Interface IO

CAN0_RX - PB4_GPIO12, peripheral MODE -5

CAN0_TX - PB5_GPIO13, peripheral MODE -5

Copyright © 2012–2019, Texas Instruments Incorporated

6.5.15.3.3 M-Boot ROM CAN Boot Mode Protocol

The CAN bootloader provides a short list of commands that are used during the application download. The description of each of these commands is given below.

Table 6-12. M-Boot ROM CAN Boot Commands

Serial No.	Command Name	Command Value	Description
1.	LM_API_UPD_PING	0x1F020000	This command is used to receive an acknowledge command from the bootloader indicating that communication has been established. This command has no data. If the device is present it will respond with an acknowledgment back to the sender.

2.	LM_API_UPD_DOWNLOAD	0x1F020040	This command sets the base address for the download as well as the size of the data to write to the device. This command should be followed by a series of LM_API_UPD_SEND_DATA that send the actual image to be programmed to the device. The command consists of two 32-bit values that are transferred LSB first.
			The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data that will be sent.
			The format of the command is as follows:
			unsigned char ucData[8];
			ucData[0] = Download Address [7:0];
			ucData[1] = Download Address [15:8];
			ucData[2] = Download Address [23:16];
			ucData[3] = Download Address [31:24];
			ucData[4] = Download Size [7:0];
			ucData[5] = Download Size [15:8];
			ucData[6] = Download Size [23:16];
			ucData[7] = Download Size [31:24];

3.	LM_API_UPD_SEND_DATA	0x1F020080	This command should only follow a LM_API_UPD_DOWNLOAD command or another LM_API_UPD_SEND_DATA command when more data is needed. Consecutive send data commands automatically increment the address and continue programming from the previous location.
			The transfer size is limited to 8 bytes at a time based on the maximum size of an individual CAN transmission. The command terminates programming once the number of bytes indicated by the LM_API_UPD_DOWNLOAD command have been received. The CAN bootloader will send a LM_API_UPD_ACK in response to each send data command to allow the CAN update application to throttle the data going to the device and not overrun the bootloader with data.
			The format of the command is as follows:
			unsigned char ucData[8];
			ucData[0] = Data[0];
			ucData[1] = Data[1];
			ucData[2] = Data[2];
			ucData[3] = Data[3];
			ucData[4] = Data[4];
			ucData[5] = Data[5];
			ucData[6] = Data[6];
			ucData[8] = Data[7];

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-12. M-Boot ROM CAN Boot Commands (continued)

Serial No.	Command Name	Command Value	Description
4.	LM_API_UPD_RESET	0x1F0200C0	This command is used to tell the M-Boot ROM CAN bootloader to start the application that is just downloaded on to the RAM. The application RUN ADDRESS is same as the application DOWNLOADED address, when using this command to start the application. If the application RUN address is different from application start address, then it is recommended to use "LM_API_UPD_RUN" command instead.

5.	LM_API_UPD_RUN	0x1F020200	This command is sent to the bootloader to transfer execution control to the specified address. The command is followed by a 32-bit value, transferred MSB first, that is the address to which execution control is transferred.
			The format of the command is as follows:
			unsigned char ucCommand[5];
			ucCommand[0] = COMMAND_RUN;
			ucCommand[1] = Run Address [31:24];
			ucCommand[2] = Run Address [23:16];
			ucCommand[3] = Run Address [15:8];
			ucCommand[4] = Run Address [7:0];

6.	LM_API_UPD_ACK	0x1F020100	This message is sent from the boot laoder if it receives a valid packet. There is no DATA associated with the packet.
7.	LM_API_UPD_NACK	0x1F0201C0	This message is sent from the boot laoder if it receives an invalid packet or an unsupported command. There is no DATA associated with the packet.

The bootloader uses the PacketWrite() function in order to send a packet of data to another device. This function encapsulates all of the steps necessary to send a valid packet to another device including waiting for the acknowledge or not-acknowledge from the other device.

Received packets use the same format as sent packets. The bootloader uses the PacketRead () function in order to receive or wait for a packet from another device. This function does not take care of acknowledging or not-acknowledging the packet to the other device. This allows the contents of the packet to be checked before sending back a response.

The steps necessary to acknowledge reception of a packet are implemented in the UpdaterCAN () function. If this function detects that a valid command is received from the sender then it acknowledges the packet by sending an LM_API_UPD_ACK. If the received packet/command is not valid then the NACK packet is sent as shown in above table.

A not-acknowledge byte is sent out whenever a sent packet is detected to have an error, usually as a result of a checksum error or just malformed data in the packet. This allows the sender to re-transmit the previous packet.

The below function call sequence gives details of the flow when the CAN boot mode is selected on the device.

- ResetIsr()
- mbrom_init_device()
- mbrom_master_system_init ()
- mbrom_control_system_init()
- mbrom_analog_system_init()
- M-Boot ROM WIR Mode Check – Please refer to WIR Mode section in *System Control and Interrupts* chapter.
- mbrom_get_bootmode()
  - ConfigureCAN();
    - UpdaterCAN();

Copyright © 2012–2019, Texas Instruments Incorporated

- mbrom_start_app(M_BOOT_ROM_CAN_BOOT_MODE_ENTRY_POINT)

6.5.15.4 M-Boot ROM Parallel IO Boot Mode

This section gives details on the Parallel IO boot mode in M-Boot ROM on these devices.

6.5.15.4.1 M-Boot ROM Parallel Boot Mode IO Configuration

Below are the IOs used in parallel boot mode.

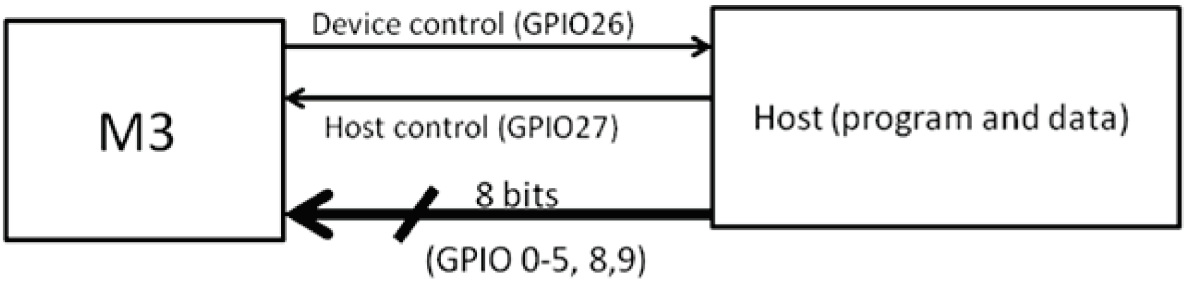
D0- PA0_GPIO0, GPIO mode
D1- PA1_GPIO1, GPIO mode
D2- PA2_GPIO2, GPIO mode
D3- PA3_GPIO3, GPIO mode
D4- PA4_GPIO4, GPIO mode
D5- PA5_GPIO5, GPIO mode
D6- PB0_GPIO8, GPIO mode
D7- PB1_GPIO9, GPIO mode

HOST_CTRL – PE3_GPIO27, GPIO mode
DSP_CTRL – PE2_GPIO26, GPIO mode

6.5.15.4.2 M-Boot ROM Parallel Boot Mode Protocol

The parallel general purpose I/O (GPIO) boot mode asynchronously transfers code from GPIO0 -GPIO5, GPIO8-GPIO9 to internal memory. Each value is 8 bits long and follows the same data flow as outlined in Section 6.6.14.1. The exception is that the C2000 hex utility hex file generated for the master subsystem boot has the least significant byte (LSB) and the most significant byte (MSB) swapped when compared to the hex file generated for the control subsystem boot. The rest of the concepts for data format and data flow are similar between the master subsystem and control subsystem.

Figure 6-5. Overview of Parallel GPIO Bootloader Operation



The master subsystem communicates with the external host device by polling/driving the GPIO27 and GPIO26 lines. The handshake protocol shown in Figure 6-6 must be used to successfully transfer each word via GPIO [9,8,5:0]. This protocol is very robust and allows for a slower or faster host to communicate with the master subsystem.

Two consecutive 8-bit words are read to form a single 16-bit word. The most significant byte (MSB) is read first followed by the least significant byte (LSB). In this case, data is read from GPIO[9,8,5:0].

The 8-bit data stream is shown in Table 6-13.

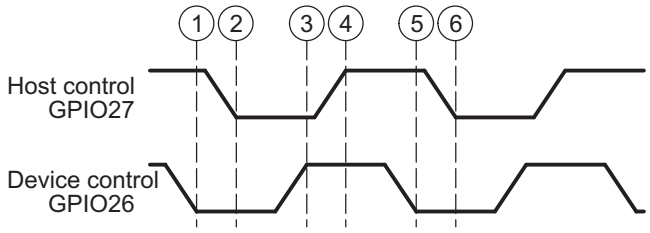
Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-13. Parallel GPIO Boot 8-Bit Data Stream

Bytes		GPIO[9,8,5:0] (Byte 1 of 2)	GPIO[9,8,5:0] (Byte 2 of 2)	Description
1	2	08	AA	0x08AA (KeyValue for memory width = 16bits)
3	4	00	00	8 reserved words (words 2 - 9)
...	...	...	...	...
17	18	00	00	Last reserved word
19	20	00	BB	Entry point PC[22:16]
21	22	CC	DD	Entry point PC[15:0] (PC = 0x00BBCCDD)
23	24	MM	NN	Block size of the first block of data to load = 0xMMNN words
25	26	AA	BB	Destination address of first block Addr[31:16]
27	28	CC	DD	Destination address of first block Addr[15:0] (Addr = 0xAABBCCDD)
29	30	AA	BB	First word of the first block in the source being loaded = 0xAABB
...				...
...				Data for this section.
				...
.		AA	BB	Last word of the first block of the source being loaded = 0xAABB
.		MM	NN	Block size of the 2nd block to load = 0xMMNN words
.		AA	BB	Destination address of second block Addr[31:16]
.		CC	DD	Destination address of second block Addr[15:0]
.		AA	BB	First word of the second block in the source being loaded
.				…
n	n+1	AA	BB	Last word of the last block of the source being loaded (More sections if required)
n+2	n+3	00	00	Block size of 0000h - indicates end of the source program

The device first signals the host that it is ready to begin data transfer by pulling the GPIO26 pin low. The host load then initiates the data transfer by pulling the GPIO27 pin low. The complete protocol is shown in the diagram below:

Figure 6-6. Parallel GPIO Bootloader Handshake Protocol



1. The device indicates it is ready to start receiving data by pulling the GPI2O6 pin low.
2. The bootloader waits until the host puts data on GPIO [9,8,5:0]. The host signals to the device that data is ready by pulling the GPIO27 pin low.
3. The device reads the data and signals the host that the read is complete by pulling GPIO26 high.
4. The bootloader waits until the host acknowledges the device by pulling GPIO27 high.
5. The device again indicates it is ready for more data by pulling the GPIO26 pin low.

This process is repeated for each data value to be sent.

Figure 6-7 shows an overview of the Parallel GPIO bootloader flow.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-7. Parallel GPIO Mode Overview

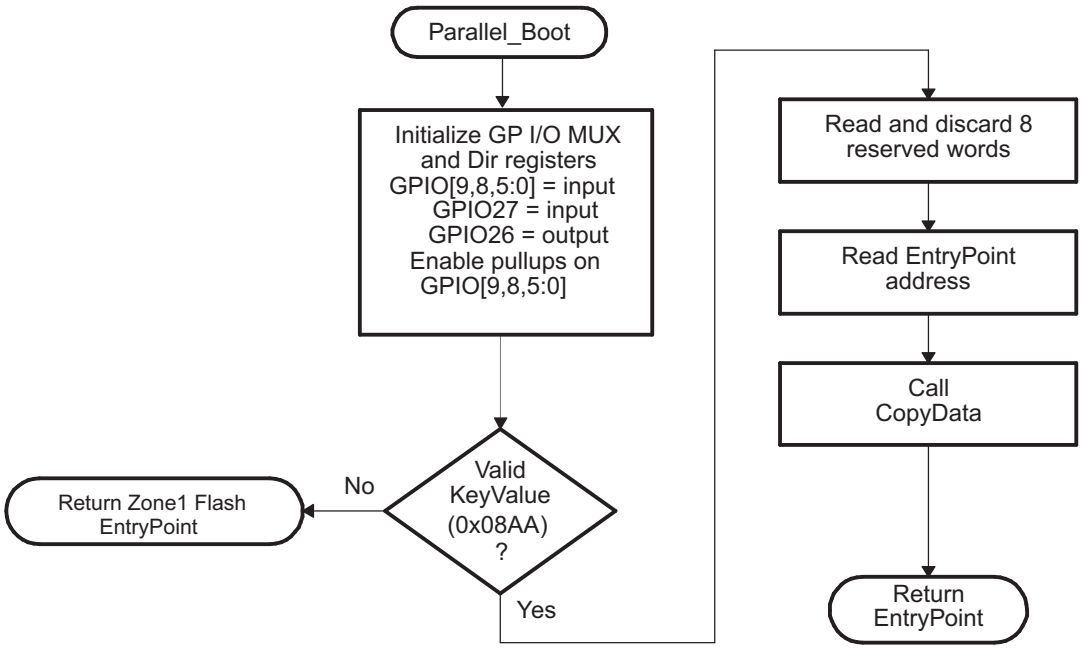


Figure 6-8 shows the transfer flow from the host side. The operating speed of the CPU and host are not critical in this mode as the host will wait for the device and the device will in turn wait for the host. In this manner the protocol will work with both a host running faster and a host running slower than the device.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-8. Parallel GPIO Mode - Host Transfer Flow

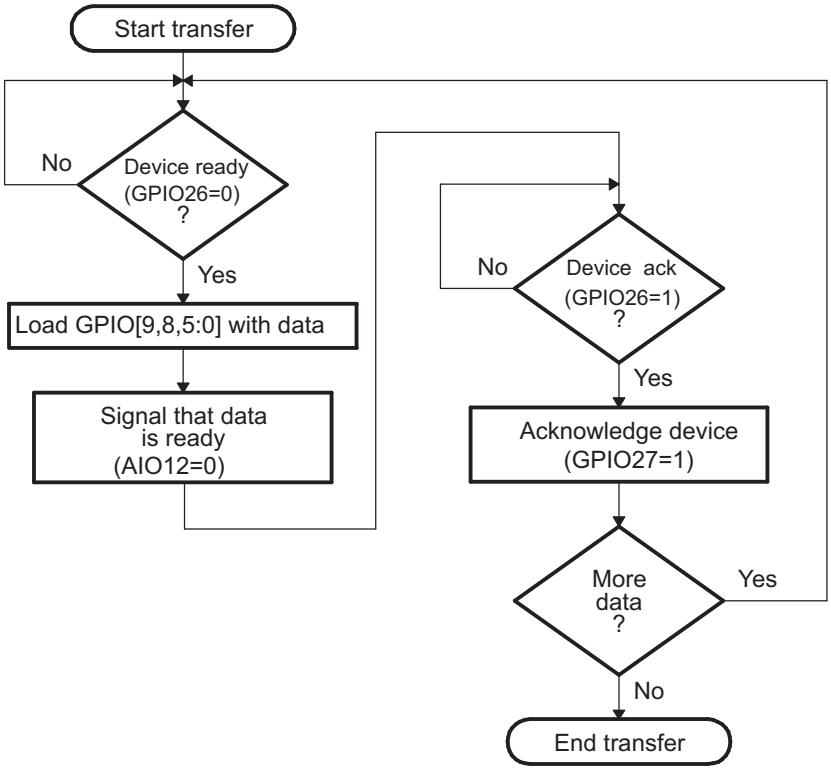


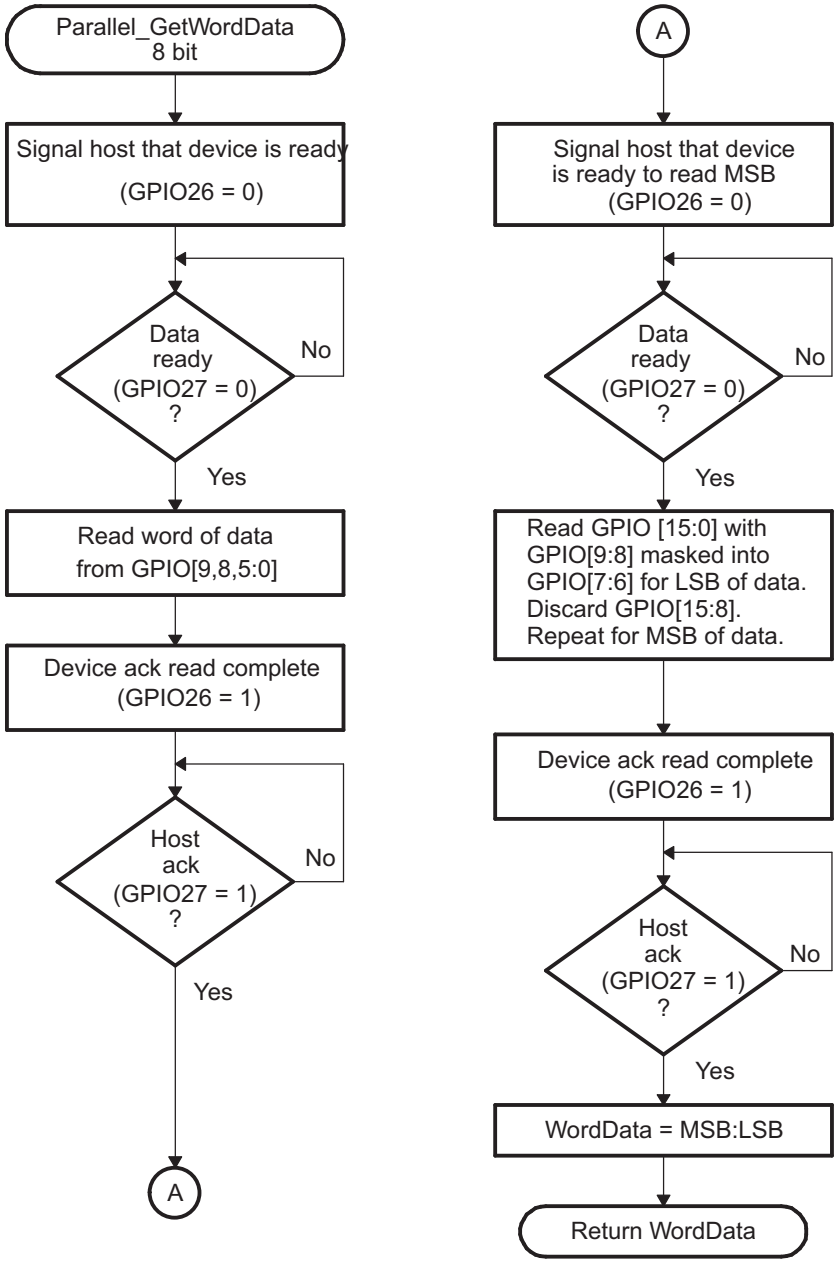
Figure 6-9 shows the flow used to read a single word of data from the parallel port.

- **8-bit data stream**

  The 8-bit routine, shown in Figure 6-9, shows how the least significant byte (LSB) of the word is to be fetched. The routine will then perform a second read to fetch the most significant byte (MSB). It then combines the MSB and LSB into a single 16-bit value to be passed back to the calling routine.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-9. 8-Bit Parallel GetWord Function



The below function call sequence gives details of the flow when Parallel boot mode is selected on the device.

- ResetIsr()
- mbrom_init_device()
- mbrom_master_system_init ()
- mbrom_control_system_init()
- mbrom_analog_system_init()
- M-Boot ROM WIR Mode Check – Please refer to WIR Mode section in the *System Control and Interrupts* chapter.
- mbrom_get_bootmode()
  - o Parallel_Boot();
- mbrom_start_app(M_BOOT_ROM_PARALLEL_BOOT_MODE_ENTRY_POINT)

Copyright © 2012–2019, Texas Instruments Incorporated

6.5.15.5 M-Boot ROM Flash Boot Mode

If this boot mode is selected, Boot ROM branches is the flash entry point as explained in Section 6.5.7.1.

The function call sequence below gives details of the flow when Flash boot mode is selected on the device.

- ResetIsr()
- mbrom_init_device()
- mbrom_master_system_init ()
- mbrom_control_system_init()
- mbrom_analog_system_init()
- M-Boot ROM WIR Mode Check – Please refer to WIR Mode section in the *System Control and Interrupts* chapter.
- mbrom_get_bootmode()
- mbrom_start_app(M_BOOT_ROM_FLASH_ENTRY_POINT)

6.5.15.6 M-Boot ROM RAM Boot Mode

If this boot mode is selected Boot ROM branches is the RAM entry point as explained in Section 6.5.7.2.

The function call sequence below gives details of the flow when RAM boot mode is selected on the device.

- ResetIsr()
- mbrom_init_device()
- mbrom_master_system_init ()
- mbrom_control_system_init()
- mbrom_analog_system_init()
- M-Boot ROM WIR Mode Check – Please refer to WIR Mode section in the *System Control and Interrupts* chapter.
- mbrom_get_bootmode()
- mbrom_start_app(M_BOOT_ROM_RAM_ENTRY_POINT)

6.5.15.7 M-Boot ROM SSI0 Master Boot Mode

In this mode, M-Boot ROM configures the SSI0 peripheral as master and tries to boot from a SSI slave.

The boot protocol used in this mode is exactly the same as the C-Boot ROM SPI Master boot mode explained in Section 6.6.14.5. The data transfer procedure is explained in Section 6.6.14.2 and the data copy procedure is explained in Section 6.6.14.3 under the C-Boot ROM section. The SSI slave is expected to send the data packets in the same format as defined in M-Boot ROM Parallel boot mode in Table 6-30.

6.5.15.8 M-Boot ROM I2C0 Master Boot Mode

In this mode, M-Boot ROM configures the I2C0 peripheral as the master and tries to boot from an I2C slave at address 0x52.

The boot protocol used in this mode is exactly the same as the C-Boot ROM I2C Master boot mode explained in the C-Boot ROM Boot I2C Boot Mode section. The data transfer procedure is explained in Section 6.6.14.2 and data copy procedure is explained in Section 6.6.14.3 under the C-Boot ROM section.

The I2C slave is expected to send the data packets in the same format as defined in M-Boot ROM Parallel boot mode in Table 6-30.

By default, the main clock frequency is taken as 20 MHz for divider calculation for I2C Fast mode (400 Kbps) operation. The main clock frequency can be use-defined by programming OTP location 0x68081C. If the user wishes to lock the main clock frequency at 20 MHz, then the OTP location must be programmed with 0x00000000.

Copyright © 2012–2019, Texas Instruments Incorporated

6.5.15.9 M-Boot ROM OTP Boot Mode

If this boot mode is selected, Boot ROM branches to the OTP entry point as explained in Section 6.5.7.9. The function call sequence below gives details of the flow when OTP boot mode is selected on the device.

- ResetIsr()
- mbrom_init_device()
- mbrom_master_system_init ()
- mbrom_control_system_init()
- mbrom_analog_system_init()
- M-Boot ROM WIR Mode Check – Please refer to WIR Mode section in the _System Control and Interrupts _chapter.
- mbrom_get_bootmode()
- mbrom_start_app(M_BOOT_ROM_OTP_ENTRY_POINT)

6.6 C-Boot ROM Description

The sequence followed by the control subsystem boot ROM is that it has to initialize the C28x CPU core, initialize PIE to handle IPC commands from master and put C28x CPU in IDLE Low Power mode.

C-Boot ROM wakes up on an IPC interrupt from master subsystem, serves the IPC command and goes back to IDLE or start the control subsystem application as commanded by the master.

6.6.1 C-Boot ROM Memory Map

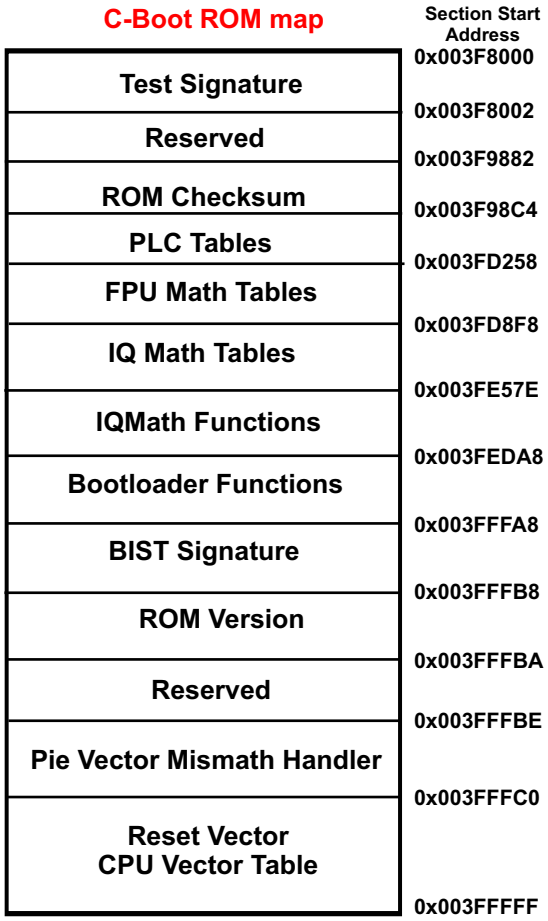
The boot ROM is an 32K x 16 block of read-only memory located at addresses 0x3F 8000 - 0x3F FFFF.

The on-chip boot ROM is factory programmed with boot-load routines and both fixed-point and floating-point math tables. These are for use with the C28x™ IQMath Library - A Virtual Floating Point Engine (SPRC087) and the C28x FPU Fast RTS Library (SPRC664).

Figure 6-10 shows the memory map of the on-chip boot ROM. The memory block is 32Kx16 in size and is located at 0x3F 8000 - 0x3F FFFF in both program and data space.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-10. C-Boot ROM Memory Map



6.6.1.1 On-Chip C-Boot ROM Math Tables

Approximately 4K of the boot ROM is reserved for floating-point and IQ math tables. These tables are provided to help improve performance and save SARAM space.

The floating-point math tables included in the boot ROM are used by the Texas Instruments™ C28x FPU Fast RTS Library (SPRC664). The C28x Fast RTS Library is a collection of optimized floating-point math functions for C programmers of the C28x with floating-point unit. Designers of computationally intensive real-time applications can achieve execution speeds considerably faster than what are currently available without having to rewrite existing code. The functions listed in the features section are specifically optimized for the C28x + FPU controllers. The Fast RTS library accesses the floating-point tables through the FPUmathTables memory section. If you do not wish to load a copy of these tables into the device, use the boot ROM memory addresses and label the section as “NOLOAD” as shown in Example 6-1. This facilitates referencing the look-up tables without actually loading the section to the target.

The following math tables are included in the boot ROM:

- **Sine/Cosine Table, Single-precision Floating point**
  - Table size: 1282 words
  - Contents: 32-bit floating-point samples for one and a quarter period sine wave
- **Normalized Arctan Table, Single-precision Floating point**
  - Table size: 388 words
  - Contents 32-bit second order coefficients for line of best fit
- **Exp Coefficient Table, Single-precision Floating point**
  - Table size: 20 words

Copyright © 2012–2019, Texas Instruments Incorporated

– Contents: 32-bit coefficients for calculating exp (X) using a Taylor series

Example 6-1. Linker Command File to Access FPU Tables

```
MEMORY
{
   PAGE 0 :
   ...
   FPUTABLES   : origin = 0x3FD860, length = 0x0006A0
   ...
}
SECTIONS
{
   ...
   FPUmathTables :  > FPUTABLES, PAGE = 0, TYPE = NOLOAD,
   ...
}
```

The fixed-point math tables included in the boot ROM are used by the Texas Instruments™ *C28x™ IQMath Library - A Virtual Floating Point Engine* (SPRC087). The 28x IQmath Library is a collection of highly optimized and high precision mathematical functions for C/C++ programmers to seamlessly port a floating-point algorithm into fixed-point code on TMS320C28x devices.

These routines are typically used in computational-intensive real-time applications where optimal execution speed and high accuracy is critical. By using these routines, you can achieve execution speeds that are considerably faster than equivalent code written in standard ANSI C language. In addition, by providing ready-to-use high precision functions, the TI IQmath Library can shorten significantly your DSP application development time.

The IQmath library accesses the tables through the IQmathTables and the IQmathTablesRam linker sections. The IQmathTables section is completely included in the boot ROM. From the IQmathTablesRam section, only the IQexp table is included and the remainder must be loaded into the device if used. If you do not wish to load a copy of these tables already included in the ROM into the device, use the boot ROM memory addresses and label the sections as “NOLOAD” as shown in Example 6-2. This facilitates referencing the look-up tables without actually loading the section to the target. Refer to the IQMath Library documentation for more information.

Example 6-2. Linker Command File to Access IQ Tables

```
MEMORY
{
   PAGE 0 :
   ...
   IQTABLES  (R)  : origin = 0x3FDF00, length = 0x000B50
   IQTABLES2 (R)  : origin = 0x3FEA50, length = 0x00008C
   IQTABLES3 (R)  : origin = 0x3FEADC, length = 0x0000AA
...
}
SECTIONS
{
   ...
   IQmathTables :  load = IQTABLES, type = NOLOAD, PAGE = 0
   IQmathTables2 > IQTABLES2, type = NOLOAD, PAGE = 0

   {
       IQmath.lib<IQNexpTable.obj> (IQmathTablesRam)
   }
       IQmathTables3 : load = IQTABLES3, PAGE = 0
   {
       IQNasinTable.obj (IQmathTablesRam)
   }

       IQmathTablesRam : load = DRAML1, PAGE = 1
```


Copyright © 2012–2019, Texas Instruments Incorporated

Example 6-2. Linker Command File to Access IQ Tables (continued)

```
        ...
}
```

The following math tables are included in the Boot ROM:

- **Sine/Cosine Table, IQ Math Table**
  - Table size: 1282 words
  - Q format: Q30
  - Contents: 32-bit samples for one and a quarter period sine wave

  This is useful for accurate sine wave generation and 32-bit FFTs. This can also be used for 16-bit math, just skip over every second value.
- **Normalized Inverse Table, IQ Math Table**
  - Table size: 528 words
  - Q format: Q29
  - Contents: 32-bit normalized inverse samples plus saturation limits

  This table is used as an initial estimate in the Newton-Raphson inverse algorithm. By using a more accurate estimate the convergence is quicker and hence cycle time is faster.
- **Normalized Square Root Table, IQ Math Table**
  - Table size: 274 words
  - Q format: Q30
  - Contents: 32-bit normalized inverse square root samples plus saturation

  This table is used as an initial estimate in the Newton-Raphson square-root algorithm. By using a more accurate estimate the convergence is quicker and hence cycle time is faster.
- **Normalized Arctan Table, IQ Math Table**
  - Table size: 452 words
  - Q format: Q30
  - Contents 32-bit second order coefficients for line of best fit plus normalization table

  This table is used as an initial estimate in the Arctan iterative algorithm. By using a more accurate estimate the convergence is quicker and hence cycle time is faster.
- **Rounding and Saturation Table, IQ Math Table**
  - Table size: 360 words
  - Q format: Q30
  - Contents: 32-bit rounding and saturation limits for various Q values
- **Exp Min/Max Table, IQMath Table**
  - Table size: 120 words
  - Q format: Q1 - Q30
  - Contents: 32-bit Min and Max values for each Q value
- **Exp Coefficient Table, IQMath Table**
  - Table size: 20 words
  - Q format: Q31
  - Contents: 32-bit coefficients for calculating exp (X) using a taylor series
- **Inverse Sin/Cos Table, IQ Math Table**
  - Table size: 85 x 16
  - Q format: Q29
  - Contents: Coefficient table to calculate the formula f(x) = c4*x^4 + c3*x^3 + c2*x^2 + c1*x + c0.

Copyright © 2012–2019, Texas Instruments Incorporated

6.6.1.2 On-Chip C-Boot ROM IQmath Functions

The following IQmath functions are included in the Boot ROM:

- IQNatan2 N= 15, 20, 24, 29
- IQNcos N= 15, 20, 24, 29
- IQNdiv N= 15, 20, 24, 29
- IQisqrt N= 15, 20, 24, 29
- IQNmag N= 15, 20, 24, 29
- IQNsin N= 15, 20, 24, 29
- IQNsqrt N= 15, 20, 24, 29

These functions can be accessed using the IQmath boot ROM symbol library included with the boot ROM source. If this library is linked in the project before the IQmath library, and the linker-priority option is used, then any math tables and IQmath functions within the boot ROM will be used first. Refer to the *IQMath Library* documentation for more information.

6.6.1.3 C-Boot ROM Version and Checksum

C-Boot ROM contains its own version number located at 0x003F FFB8, as shown in Table 6-14. This version is incremented each time C-Boot ROM code is modified. The next word contains the month and year (MM/YY in decimal) that the boot code was released. The next four words contain a checksum value for C-Boot ROM. Taking a 64-bit summation of all addresses within the ROM, except for the checksum locations, generates this checksum.

Table 6-14. C-Boot ROM Version and Checksum Information

C-Boot ROM Address	Contents
0x003FFFB8	Revision No = 0x0100 (C-BOOTROM version number)
0x003FFFB9	Release Date = 0x0211 (02/2011)
0x3F9882 - 0x3F98C4	Checksum

6.6.1.4 C-Boot ROM PIE Vector Mismatch Handler

As explained in the Control Subsystem PIE Vector Address Validity Check section of the *System Control and Interrupts* chapter, whenever the control subsystem detects that PIE Vector address of the current interrupt vector fetch or current NMI vector fetch is corrupted the CPU program control will jump to location 0x003FFFBE and starts executing code at that location. As shown in the memory map and below table C-BootROM has a branch instruction installed at this 32-bit location to handle the PIE Vector mismatch event. This branch instruction branches to cbrom_pie_vect_mismatch_handler function. Please refer to Section 6.6.12.2 and Section 6.6.1.3 for more details on what this function does whenever it is executed.

Table 6-15. C-Boot ROM PIE Mismatch Handler

C-Boot ROM Address	Contents
0x003FFFBE – 0x003FFFBF	BF _cbrom_pie_vect_mismatch_handler, UNC

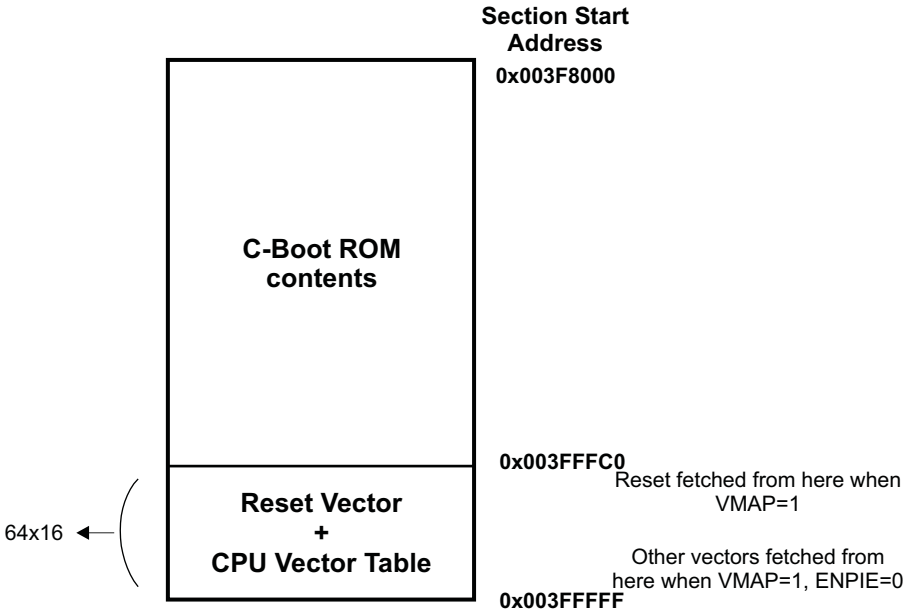
Note: This function will be called when the control subsystem detects a mismatch for both normal vector fetch and NMI vector fetch. The master subsystem will receive and NMI if the mismatch on the control subsystem happens during an NMI vector fetch. M-Boot ROM handling of this NMI on the master subsystem is explained in the M-Boot ROM chapter. The master subsystem will not receive any NMI if the mismatch on the control subsystem is while doing a normal PIE interrupt vector fetch, in which case C-Boot ROM sends an IPC message to the master subsystem as shown in Section 6.6.12.2 and Section 6.6.1.3.

Copyright © 2012–2019, Texas Instruments Incorporated

6.6.1.5 C-Boot ROM Vector Table

A CPU vector table resides in boot ROM memory from address 0x3F FFC0 - 0x3F FFFF. This vector table is active after reset when VMAP = 1, ENPIE = 0 (PIE vector table disabled).

Figure 6-11. C-Boot ROM Vector Table Map



A The VMAP bit is located in Status Register 1 (ST1). VMAP is always 1 on reset. It can be changed after reset by software, however, the normal operating mode will be to leave VMAP=1.

B The ENPIE bit is located in the PIECTRL register. The default state of this bit at reset is 0, which disables the Peripheral Interrupt Expansion block (PIE).

The only vector that will normally be handled from the internal boot ROM memory is the reset vector located at 0x3F FFC0. The reset vector is factory programmed to point to the InitBoot function stored in the boot ROM. This function starts the boot load process.

PIE and all the peripheral interrupts are disabled by default on reset on the control subsystem. ITRAP and NMI exceptions are enabled by default, so until PIE is enabled, the control subsystem CPU will fetch for interrupt vector from the CPU vector table in C-Boot ROM for NMI, ITRAP or Reset exceptions as shown in Table 6-16

For TI silicon debug and test purposes the interrupt vectors (not the NMI and ITRAP exceptions) located in the boot ROM memory point to locations in the M0 SARAM block as described in below table. During silicon debug, user can program the specified locations in M0 with branch instructions to catch any vectors fetched from boot ROM. This is not required for normal device operation.

Table 6-16. C-Boot ROM CPU Vector Table

Vector Name (Number)	Vector Address or Location in BootROM – When PIE is disabled	Contents (Handler address)
RESET	0x003FFFC0	InitBoot
INT1	0x003FFFC2	0x00000042
INT2	0x003FFFC4	0x00000044
INT3	0x003FFFC6	0x00000046
INT4	0x003FFFC8	0x00000048
INT5	0x003FFFCA	0x0000004A
INT6	0x003FFFCC	0x0000004C
INT7	0x003FFFCE	0x0000004E
INT8	0x003FFFD0	0x00000050

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-16. C-Boot ROM CPU Vector Table (continued)

Vector Name (Number)	Vector Address or Location in BootROM – When PIE is disabled	Contents (Handler address)
INT9	0x003FFFD2	0x00000052
INT10	0x003FFFD4	0x00000054
INT11	0x003FFFD6	0x00000056
INT12	0x003FFFD8	0x00000058
INT13	0x003FFFDA	0x0000005A
INT14	0x003FFFDC	0x0000005C
DLOGINT	0x003FFFDE	0x0000005E
RTOSINT	0x003FFFE0	0x00000060
Reserved	0x003FFFE2	0x00000062
NMI	0x003FFFE4	cbrom_handle_nmi
ITRAP or ILLEGAL	0x003FFFE6	cbrom_itrap_isr
USER1	0x003FFFE8	0x00000068
USER2	0x003FFFEA	0x0000006A
USER3	0x003FFFEC	0x0000006C
USER4	0x003FFFEE	0x0000006E
USER5	0x003FFFF0	0x00000070
USER6	0x003FFFF2	0x00000072
USER7	0x003FFFF4	0x00000074
USER8	0x003FFFF6	0x00000076
USER9	0x003FFFF8	0x00000078
USER10	0x003FFFFA	0x0000007A
USER11	0x003FFFFC	0x0000007C
USER12	0x003FFFFE	0x0000007E

Note: Each of the functions in bold above are explained in detail further below in the document.

C-BootROM eventually enables PIE to handle IPC communication or commands from the master subsystem during the boot process, as will be explained further below in this chapter. After PIE is enabled the NMI, ITRAP and any enabled peripheral interrupt vectors are fetched from the PIE Vector table as shown in table 0.13 below. C-Boot ROM enables only one PIE interrupt – MTOCIPCINT1, which is interrupt line 1 in PIE GROUP 11. In other words C-Boot ROM enables PIE INT11.1 interrupt and enables PIE during the boot process. All the remaining PIE interrupts are disabled, however PIE Vector table is initialized with “cbrom_pie_isr_not_supported” handler for all the other PIE interrupts.

The table below only shows the interrupt and exception vectors that are fetched from C-Boot ROM

Please refer to the Control Subsystem PIE section in the *System Control and Interrupts* chapter for more details on PIE. And PIE group vectors, the below table only shows how C-BootROM initialized PIE Vector table during boot, user applications are free to disable PIE and re-initialize PIE as needed after the application starts.

Table 6-17. PIE Vector Table in C-Boot ROM

Vector Name (Number)	Vector Address or Location – When PIE is enabled, during boot process	Contents (Handler address)
RESET (0)	0x00000D00	cbrom_pie_isr_not_supported (Reset is always fetched from 0x3FFFC0)
INT1 (1)	0x00000D02	cbrom_pie_isr_not_supported
INT2 (2) – INT14 (14)	0x00000D04 – 0x00000D1C	cbrom_pie_isr_not_supported
DLOGINT (15)	0x00000D1E	cbrom_pie_isr_not_supported
RTOSINT (16)	0x00000D20	cbrom_pie_isr_not_supported
EMUINT (17)	0x00000D22	cbrom_pie_isr_not_supported

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-17. PIE Vector Table in C-Boot ROM (continued)

Vector Name (Number)	Vector Address or Location – When PIE is enabled, during boot process	Contents (Handler address)
NMI (18)	0x00000D24	cbrom_handle_nmi
ILLEGAL or ITRAP (19)	0x00000D26	cbrom_itrap_isr
USER1 (20) – USER12 (31)	0x00000D28 - 0x00000D3E	cbrom_pie_isr_not_supported
PIE-GROUP1		
INT1.1 (32) – INT1.8 (39)	0x00000D40 - 0x00000D4E	cbrom_pie_isr_not_supported
PIE-GROUP2		
INT2.1 (40) – INT1.8 (47)	0x00000D50 - 0x00000D5E	cbrom_pie_isr_not_supported
PIE-GROUP3		
INT3.1 (48) – INT3.8 (55)	0x00000D60 - 0x00000D6E	cbrom_pie_isr_not_supported
PIE-GROUP4		
INT4.1 (56) – INT4.8 (63)	0x00000D70 - 0x00000D7E	cbrom_pie_isr_not_supported
PIE-GROUP5		
INT5.1 (64) – INT5.8 (71)	0x00000D80 - 0x00000D8E	cbrom_pie_isr_not_supported
PIE-GROUP6		
INT6.1 (72) – INT6.8 (79)	0x00000D90 - 0x00000D9E	cbrom_pie_isr_not_supported
PIE-GROUP7		
INT7.1 (80) – INT7.8 (87)	0x00000DA0 - 0x00000DAE	cbrom_pie_isr_not_supported
PIE-GROUP8		
INT8.1 (88) – INT8.8 (95)	0x00000DB0 - 0x00000DBE	cbrom_pie_isr_not_supported
PIE-GROUP9		
INT9.1 (96) – INT9.8 (103)	0x00000DC0 - 0x00000DCE	cbrom_pie_isr_not_supported
PIE-GROUP10		
INT10.1 (104) – INT10.8 (111)	0x00000DD0 - 0x00000DDE	cbrom_pie_isr_not_supported
PIE-GROUP11		
INT11.1 (112)	0x00000DE0	cbrom_mtoc_ipc_int1_isr
INT11.2 (113) - INT11.4 (115)	0x00000DE2 - 0x00000DE6	cbrom_pie_isr_not_supported
INT11.5 (116)	0x00000DE8	TI-RESERVED
INT11.6 (117) – INT11.8 (119)	0x00000DEA - 0x00000DEE	cbrom_pie_isr_not_supported
PIE-GROUP12		
INT12.1 (120) – INT12.8 (127)	0x00000DF0 - 0x00000DFE	cbrom_pie_isr_not_supported

6.6.2 C-Boot ROM RAM Initialization

As mentioned earlier, RAM memories on the master and control subsystems on these devices must be zero-initialized before using the RAM for the first time to avoid any ECC and Parity errors. Refer to the *Internal Memory* chapter for more details on RAM ECC and Parity.

C-Boot ROM zero-initializes all the Control subsystem RAMs as part of the device initialization process if the POR reset cause bit is set in the CRESC register. For all the other resets, C-Boot ROM stack memory is zero-initialized. C-Boot ROM actions for possible reset causes are further defined below in this chapter.

6.6.3 C-Boot ROM RAM Usage

The M0 memory block address range 0x0002 - 0x01A0 is reserved for the stack and .ebss code sections during the bootload process. If code is bootloaded into this region there is no error checking to prevent it from corrupting the boot ROM stack.

Copyright © 2012–2019, Texas Instruments Incorporated

6.6.4 C-Boot ROM Boot Modes

Unlike the master subsystem, the control subsystem doesn't support automatic detection of boot modes using boot mode GPIO, because on Concerto it is up to the master subsystem application on how it wants to boot the control subsystem. C-Boot ROM supports boot using IPC commands. The master subsystem application can send certain IPC commands to C-Boot ROM to let the control subsystem boot. Table 6-18 shows different boot options supported by C-Boot ROM. These boot options programmed into the MTOCIPCBOOTMODE register should be used in conjunction with the "IPC_MTOC_EXECUTE_BOOTMODE_CMD" (value = 0x00000013) command in the MTOCIPCCOM register as detailed in the next section.

Table 6-18. C-Boot ROM Boot Modes

C-Boot ROM Boot Mode	MTOCIPCBOOTMODE Register Value	Description
BOOT_FROM_RAM	0x00000001	On receiving this command from Master subsystem, C-Boot ROM will branch to the control subsystem RAM entry point location and starts executing code from there.
BOOT_FROM_FLASH	0x00000002	On receiving this command, C-Boot ROM will branch to the control subsystem FLASH entry point and starts executing code from there.
BOOT_FROM_SCI	0x00000003	On receiving this command C-Boot ROM boots from the control subsystem SCI peripheral.
BOOT_FROM_SPI	0x00000004	On receiving this command C-Boot ROM boots from the control subsystem SPI interface.
BOOT_FROM_I2C	0x00000005	On receiving this command, C-Boot ROM boots from the control subsystem I2C interface
BOOT_FROM_PARALLEL	0x00000006	On receiving this command, C-Boot ROM boots from the control subsystem GPIO

The master subsystem application must follow the below procedure to let the control subsystem boot as commanded by master.

6.6.4.1 Master Application Procedure to Send Boot Mode IPC Commands to C-Boot ROM

The set of MTOCIPC registers mentioned in IPC section of the *System Control and Interrupts* chapter are used to communicate and give commands to C-Boot ROM. The list of IPC commands supported by C-Boot ROM is given in Section 6.6.9.3. This section explains how these IPC registers should be used by a master subsystem application to boot the control subsystem.

1. Check if C-Boot ROM is ready to accept IPC commands. If yes, then proceed to step 2 below, else, wait. CTOMIPCBOOTSTS register is used by C-Boot ROM to communicate boot status to the master subsystem. This will be explained in more detail in below sections of the document.
2. Configure the control subsystem C28x CPU as the owner for respective GPIO based on boot mode using core select feature on the GPIO. In other words assign the required GPIO for boot to the C28x core. Refer to Table 6-20 for more details on GPIO for each boot mode.
3. MTOCIPCOM = MASTER_IPC_MTOC_EXECUTE_BOOTMODE_CMD
4. MTOCIPCBOOTMODE = C-Boot ROM Boot Mode command (as mentioned in Table 6-18)
5. MTOCIPCSET = 0x80000001 (set bits 31 and 0 to enable respective bits in MTOCIPCFLG register)

MASTER_IPC_MTOC_EXECUTE_BOOTMODE_CMD mentioned in step 3 above is one of the IPC commands supported by C-Boot ROM. The entire set of IPC commands supported by C-Boot ROM and the detailed flow of on how the master subsystem can send an IPC command and how C-Boot ROM handles the IPC commands is explained in detail in the subsequent sections of this chapter.

Refer to Section 6.7 for more details on this procedure.

Copyright © 2012–2019, Texas Instruments Incorporated

6.6.5 C-Boot ROM Entry Points

This section gives details about the entry point addresses for various boot modes supported by C-Boot ROM. These entry points tell C-Boot ROM where to branch to at the end of booting as per boot mode selected.

Table 6-19. C-Boot ROM Entry Point

Name	Address	Size (x4)	Description
C_BOOTROM_FLASH_ENTRY_POINT	0x13FFF0	2	Flash entry point for boot-to-flash option
C_BOOTROM_RAM_ENTRY_POINT	0x000000	2	RAM entry point for boot to RAM option
C_BOOTROM_SCI_ENTRY_POINT	N/A	2	Entry point will be provided by the user or Host system using the SCI boot mode option
C_BOOTROM_SPI_ENTRY_POINT	N/A	2	Entry point will be provided by the user or Host system using the SPI boot mode option
C_BOOTROM_I2C_ENTRY_POINT	N/A	2	Entry point will be provided by the user or Host system using the I2C boot mode option
C_BOOTROM_PARALLEL_ENTRY_POINT	N/A	3	Entry point will be provided by the user or Host system using the parallel boot mode option

6.6.5.1 C-Boot ROM Boot-to-RAM Entry Point

C-Boot ROM Ram entry point by default is fixed to 0x00000000 in M0 RAM. This location will be referred to as C_BOOTROM_RAM_ENTRY_POINT further in this document.

This means if the master subsystem application sends a boot to RAM IPC command, then C-Boot ROM branches to location 0x00000000 in M0 RAM. User applications which use this option must have their main function located at this address or have a branch to main() instruction at this location.

The boot to RAM option is mainly helpful during code development.

6.6.5.2 C-Boot ROM Boot-to-Flash Entry Point

C-Boot ROM Flash entry point by default is fixed at 0x0013FFF0, this means that whenever C-Boot ROM receive a Boot to flash boot mode IPC command from master subsystem it will branch to this location and begins executing the code. User applications which use this option must have their main function(first function to be called to begin application execution) located at this address or have a branch to main() instruction at this location.

NOTE: On a device with erased C-Flash, choosing this option will trigger an ITRAP exception to the C28x CPU. Please refer to the C-Boot ROM exceptions and interrupt handling section of this chapter for more details on how this is handled in C-Boot ROM.

6.6.5.3 C-Boot ROM SCI-Boot Entry Point

Entry point in this boot mode will be provided by the user or host system which is sending boot code to the device. Please refer to Serial Boot Data format for more details.

6.6.5.4 C-Boot ROM SPI-Boot Entry Point

Entry point in this boot mode will be provided by the user or host system which is sending boot code to the device. Please refer to SPI Boot Data format for more details.

6.6.5.5 C-Boot ROM I2C-Boot Entry Point

Entry point in this boot mode will be provided by the user or host system which is sending boot code to the device. Please refer to I2C Boot Data format for more details.

Copyright © 2012–2019, Texas Instruments Incorporated

6.6.6 C-Boot ROM GPIO Assignment for Boot Modes

Table 6-20 gives information on the GPIOs used for each boot mode on C-Boot ROM. More details on the boot mode is given further below in this document.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-20. C-Boot ROM GPIO Assignments for Boot Modes

C-Boot ROM Boot Mode	Peripheral interface	Boot Function Name for pin	Direction	GPIO(s) used	Pin Mux Assignment	
					Peripheral Mode	Core Select
SCI	SCIA	SCITXDA	Output	PF3_GPIO35	1	Control Subsystem
		SCIRXDA	Input	PF4_GPIO36	1	Control Subsystem
I2C	I2CA	SDAA	Bi-Directional	PF0_GPIO32	1	Control Subsystem
		SCLA	Output	PF1_GPIO33	1	Control Subsystem
SPI	SPIA	SPISIMOA	Output	PD0_GPIO16	1	Control Subsystem
		SPISOMIA	Input	PD1_GPIO17	1	Control Subsystem
		SPICLKA	Output	PD2_GPIO18	1	Control Subsystem
		SPISTEA	Output	PD3_GPIO19	1	Control Subsystem
Parallel Boot Mode	GPIO (s)	D0	Input	PA0_GPIO0	0(default)	Control Subsystem
		D1	Input	PA1_GPIO1	0(default)	Control Subsystem
		D2	Input	PA2_GPIO2	0(default)	Control Subsystem
		D3	Input	PA3_GPIO3	0(default)	Control Subsystem
		D4	Input	PA4_GPIO4	0(default)	Control Subsystem
		D5	Input	PA5_GPIO5	0(default)	Control Subsystem
		D6	Input	PB0_GPIO8	0(default)	Control Subsystem
		D7	Input	PB1_GPIO9	0(default)	Control Subsystem
		HOST_CTRL	Input	PE3_GPIO27	0(default)	Control Subsystem
		DSP_CTRL	Ouput	PE2_GPIO26	0(default)	Control Subsystem

6.6.7 C-Boot ROM Clock Initializations

On these devices, M-Boot ROM configures the clock settings before bringing the control subsystem out of reset. C-Boot ROM doesn't configure any clock settings.

Please refer to Section 6.5.8 and the Clock Control section of the *System Control and Interrupts* chapter for more details.

6.6.8 C-Boot ROM Functional Flow

As shown in the CPU vector table, InitBoot is the main function called whenever C-Boot ROM is executed after a reset. This section explains the flow of C-Boot ROM function-by-function.

6.6.8.1 INITBOOT ()

- Initialize the stack pointer
- Put CPU in C28x Object Mode
- If POR is the source of reset, initialize M0, M1, L0-L3 and CTOM MSGRAM to 0x0

Copyright © 2012–2019, Texas Instruments Incorporated

- If POR is not the source of reset, zero initialize stack
- Call crom_control_system_init()

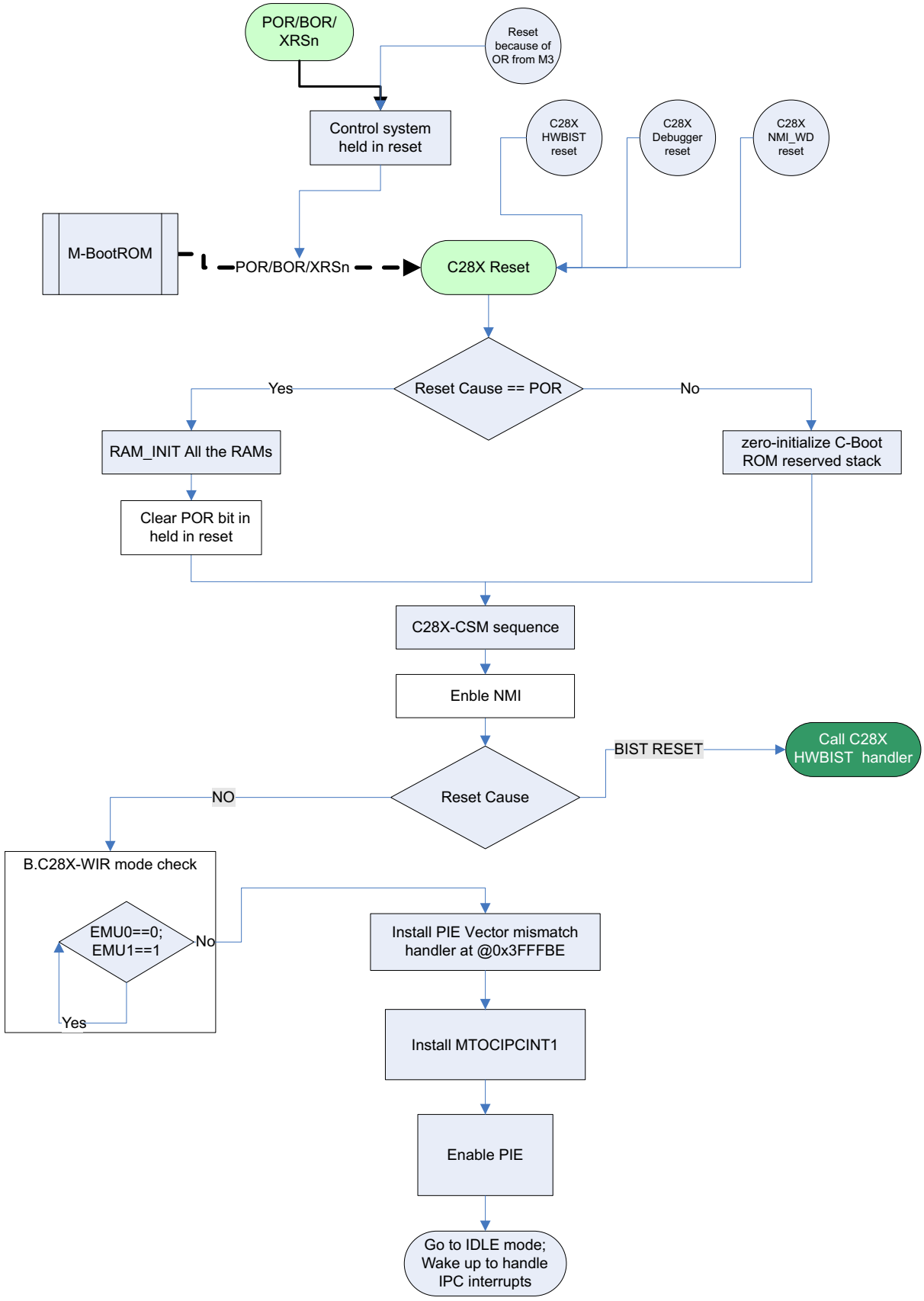
6.6.8.2 CROM_CONTROL_SYSTEM_INIT()

- Control subsystem CSM initialization sequence – Refer to the Security Module section in the *System Control and Interrupts* chapter.
- Initialize boot environment – local and global variables initialization for boot.
- C-Boot ROM WIR Mode Check – Please refer to WIR Mode section in System Control and Interrupts chapter.
- Initialize PIE
- Install MTOCIPCINT1 handler – to handle IPC commands from master.
- Install PIE NMI and ITRAP handlers.
- Enable PIEGROUP11.1 interrupt.
- Enable PIE
- Set Boot Status for Master applications to read
- Enter Low Power IDLE mode.

The C-Boot ROM flow chart is shown below.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-12. C-Boot ROM Flow Chart



Copyright © 2012–2019, Texas Instruments Incorporated

6.6.9 C-Boot ROM MTOC IPC Commands

To enable master subsystem applications control how the C28x control subsystem boots, C-Boot ROM supports IPC commands given in Table 6-21.

C-Boot ROM enables PIE to handle MTOCIPCINT1 interrupt, this interrupt will be triggered to C28x CPU of the control subsystem when bit 0 of MTOCIPCINTFLG register is set by the master subsystem. Only this interrupt is configured and handled by C-BootROM all other interrupts in PIE are disabled and/or ignored, but in case if they are triggered an IPC message as described in section 1.6.12 will be sent to the master subsystem.

Please refer to IPC section of *System Control and Interrupts* chapter for more details on IPC module registers.

The master subsystem application uses IPC between C-BootROM and master application to get the booting related data from Master system or to tell the control subsystem to fetch boot related data from its own peripherals or to tell the control subsystem boot ROM to just service the IPC commands from master.

MTOCIPCINT1 interrupt handler function looks into MTOCIPCOMMAND registers, decodes the IPC commands and if it is a valid command handles the command.

6.6.9.1 Procedure for Master Subsystem Application to Send IPC Commands to C-Boot ROM and Wait for Response

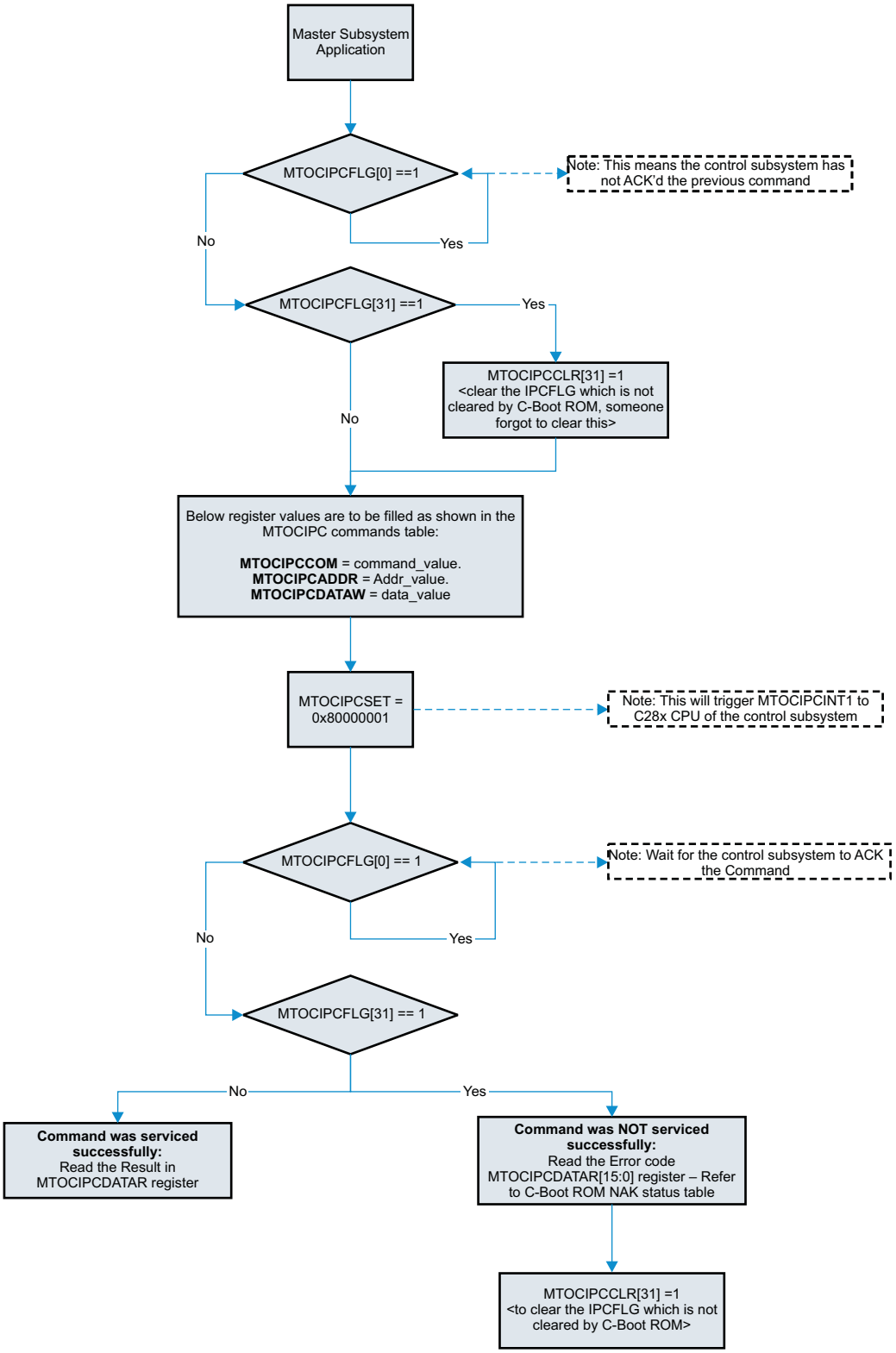
The procedure for the master subsystem application software to send an IPC command to C-Boot ROM is as below.

1. Write command to MTOCIPCCOM register
2. Write address to MTOCIPCADDR register.
3. Write data to MTOCIPCDATAW register
4. Set bit 31 in MTOCIPCSET register. C-Boot ROM will ignore the command if both bit 31 and bit 0 of MTOCIPCFLG register are not set.
5. Set bit 0 in MTOCIPCSET register. This will set bit 0 in MTOCIPCFLG register and trigger MTOCIPCINT1 to the control subsystem C28X CPU.
6. Master subsystem software will wait for MTOCIPCFLG[0] to be cleared by C-Boot ROM:
   - a. if MTOCIPCFLG[0] is cleared and MTOCIPCFLG[31] is not cleared it should understand that the command it sent was invalid or an error occurred, in this case the master subsystem application software can look at the MTOCIPCDATAR register to know more about the error occurred and why C-Boot ROM NAK'd the IPC command.
   - b. If both MTOCIPCFLG[0] and MTOCIPCFLG[31] are cleared by C-Boot ROM then master will know that the command was executed successfully and it will take MTOCIPCDATAR register into consideration as shown in Table 6-21 based on the command.
7. Master application software will have to clear MTOCIPCFLG[31] if it wasn't cleared by C-Boot ROM and only MTOCIPCFLG[0] is cleared by C-Boot ROM, in error cases.

Figure 6-13 shows the procedure to be followed by the master subsystem application.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-13. Master Subsystem Application Procedure TO Send IPC TO C-Boot ROM



6.6.9.2 Procedure Followed by C-Boot ROM to Handle IPC Commands From Master -

Copyright © 2012–2019, Texas Instruments Incorporated

CBROM_MTOC_IPC_INT1_ISR

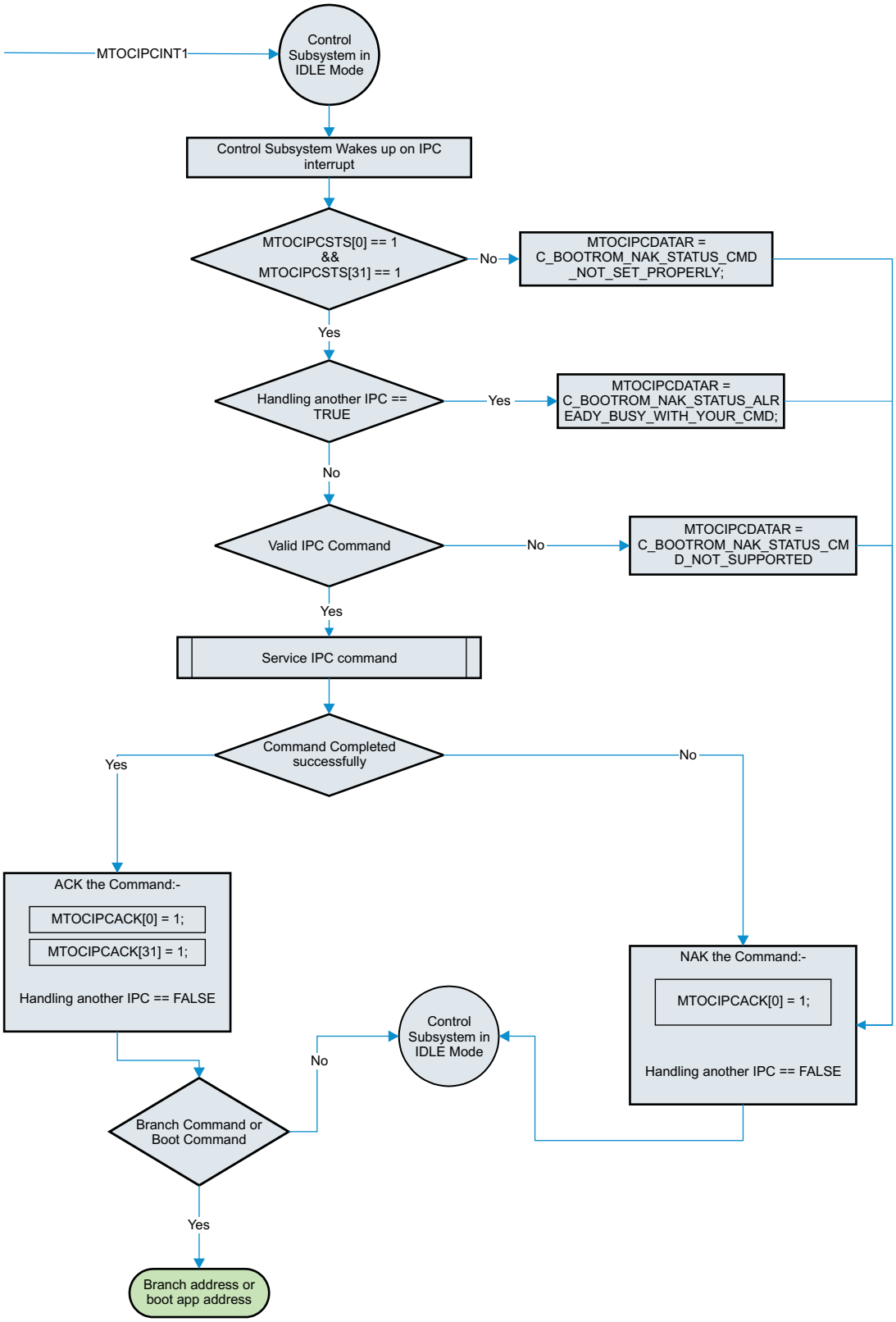
Below is the procedure that C-Boot ROM, MOTCIPCINT1 handler follows in response to IPC command from master (MTOCIPCINT1).

- Read and decode the command in MTOCIPCCOM register.
- Read/decode the address on which C-Boot ROM has to act as per the command.
- Read/decode the data with which C-Boot ROM has to act on the above address as per the command.
- Service the command.
- Clear MTOCIPCFLG[31] if the command was successfully processed, if the command was invalid or cannot execute the command because of any reason, MTOCIPCFLG[31] is not cleared and MTOCIPCDATAR register is updated with error code explained in the following sections.
- Put the result (if any) of the command in MTOCIPCDATAR register, if the command is executed successfully.
- Set bit 0 in MTOCIPCACK register. This will clear bit 0 in MTOCIPCFLG register and Master user application will know that C-Boot ROM has served the command and it can look at MTOCIPCDATAR register for the result of command.

Figure 6-14 explains how C-Boot ROM MTOCIPCINT1 handler - cbrom_mtoc_ipc_int1_isr, follows to service IPC command from Master.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-14. C-Boot ROM Handling on MTOCIPC



Copyright © 2012–2019, Texas Instruments Incorporated

6.6.9.3 MTOC IPC Command – IPC Commands Supported by C-Boot ROM

Table 6-21 shows all the IPC commands supported by C-Boot ROM. The description column shows what C-Boot ROM does to service the IPC command.

The 'Value' column shows the value that has to be written to MTOCIPCCOM register to send the command in the MTOCIPCCOM column. The MTOCIPCADDR and MTOCIPCDATAW columns show how these registers should be initialized by the master or how C-Boot ROM interprets these register contents to be – for the respective command in the MTOCIPCCOM column.

The MTOCIPCDATAR and MTOCIPCFLG[31] columns show the values set by C-Boot ROM based on the command sent by the master.

The heading ROW also shows the READ-WRITE permissions for each register for each core. If any of the commands results in a failure, then the MTOCIPCDATAR register contents tell why the command failed. This is explained in Section 6.6.9.4.

Table 6-21. MTOC IPC Commands

Value	(M3 - R/W, C28X R) MTOCIPCCOM	MTOCIPCADDR (M3 - R/W, B.C28X R)	MTOCIPCDATAW (M3 - R/W, C28X R)	MTOCIPCDATAR (M3 - R, C28X R/W)	MTOCIPCFLG[31] = ? (MTOCIPCFLAG[0] = 0)	Description
0	MASTER_IPC_MTOC_COMMAND_ILLEGAL	DON'T CARE	DON'T CARE	Section 6.6.9.4	0x01 = Command failure	Illegal command
1	MASTER_IPC_MTOC_SET_BITS_16	Address of the 16 bit location	Data in MTOCIPCDATAW[15:0]	Data read back from address after write	0x00 = Command success	*(address) |= data;
2	MASTER_IPC_MTOC_SET_BITS_32	Address of the 32 bit register	Data;	Data read back from address after write	0x00 = Command success	*(address) |= data;
3	MASTER_IPC_MTOC_CLEAR_BITS_16	Address of the 16 bit register	Data in MTOCIPCDATAW[15:0]	Data read back after write	0x00 = Command success	*(address) &= ~data;
4	MASTER_IPC_MTOC_CLEAR_BITS_32	Address of the 32 bit register	Data	Data read back after write	0x00 = Command success	*(address) &= ~data;
5	MASTER_IPC_MTOC_DATA_WRITE_16	Address of the 16 bit register	Data in MTOCIPCDATAW[15:0]	Data read back from the address after write	0x00 = Command success	*(address) = data;
6	MASTER_IPC_MTOC_DATA_WRITE_32	Address of the 32 bit register	Data	Data read back from the address after write	0x00 = Command success	*(address) = data;
7	MASTER_IPC_MTOC_DATA_READ_16	Address of the 16 bit register	DON'T CARE	Data in MTOCIPCDATAR[15:0]	0x00 = Command success	MTOCIPCDATAR[15:0] = *(address); Only 16 bit read from address
8	MASTER_IPC_MTOC_DATA_READ_32	Address of the 32 bit register	DON'T CARE	32 bit data	0x00 = Command success	MTOCIPCDATAR[31:0] = *(address); 32 bits read from address
9	MASTER_IPC_MTOC_SET_BITS_PROTECTED_16	Address of the 16 bit register	Data in MTOCIPCDATAW[15:0]	Data read back from address after write	0x00 = Command success	EALLOW; *(address) |= data; EDIS;
10	MASTER_IPC_MTOC_SET_BITS_PROTECTED_32	Address of the 32 bit register	Data;	Data read back from address after write	0x00 = Command success	EALLOW; *(address) |= data; EDIS;
11	MASTER_IPC_MTOC_CLEAR_BITS_PROTECTED_16	Address of the 16 bit register	Data in MTOCIPCDATAW[15:0]	Data read back after write	0x00 = Command success	EALLOW;
						*(address) &= ~data;
						EDIS;
12	MASTER_IPC_MTOC_CLEAR_BITS_PROTECTED_32	Address of the 32 bit register	Data	Data read back after write	0x00 = Command success	EALLOW;
						*(address) &= ~data;
						EDIS;
13	MASTER_IPC_MTOC_DATA_WRITE_PROTECTED_16	Address of the 16 bit register	Data in MTOCIPCDATAW[15:0]	Data read back from the address	0x00 = Command success	EALLOW;
						*(address) = data;
						EDIS;

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-21. MTOC IPC Commands (continued)

Value	(M3 - R/W, C28X R) MTOCIPCCOM	MTOCIPCADDR (M3 - R/W, B.C28X R)	MTOCIPCDATAW (M3 - R/W, C28X R)	MTOCIPCDATAR (M3 - R, C28X R/W)	MTOCIPCFLG[31] = ? (MTOCIPCFLAG[0] = 0)	Description
14	MASTER_IPC_MTOC_DATA_WRITE_PROTECTED_32	Address of the 32 bit register	Data	Same as above	0x00 = Command success	EALLOW; *(address) = data; EDIS;
15	MASTER_IPC_MTOC_DATA_READ_PROTECTED_16	Address of the 16 bit register	DON'T CARE	Data in MTOCIPCDATAR[15:0]	0x00 = Command success	EALLOW; MTOCIPCDATAR[15:0] = *(address); EDIS; Only 16 bit read from address
16	MASTER_IPC_MTOC_DATA_READ_PROTECTED_32	Address of the 32 bit register	Same as above	32 bit data	0x00 = Command success	EALLOW; MTOCIPCDATAR[31:0] = *(address); EDIS; 32 bits read from address
17	MASTER_IPC_MTOC_BRANCH_CALL	Address where to branch to	DON'T CARE	DON'T CARE	0x00 = Command success	C-BootROM will jump to the address in ADDR register and starts executing the code from that address; PIE will be enabled when this branch occurs, it is up to the application to disable and reload PIE interrupt handlers if it wants to. The current PIE interrupt will be acknowledged before the branch is performed.
18	MASTER_IPC_MTOC_FUNCTION_CALL	Address of the function	Parameter for the function call	Result of function call (return value if any from function)	0x00 = Command success	C-BootROM will jump to the address in ADDR register and starts executing the code from that address; Data in DATAW register can be used as parameter to the function call. C-BootROM returns back to where it was after servicing the function call. Function call is performed from inside the interrupt service routine on the control subsystem so user has to keep this in mind.
19	MASTER_IPC_MTOC_EXECUTE_BOOTMODE_CMD	DON'T CARE	DON'T CARE	DON'T CARE	0x00 = Command success	Execute loaders as per requested value in the MTOCBOOTMODE register Refer to Table 6-18. These loaders are called from within the interrupt service routine context. So, any PIE interrupts of same or lower level will be pended until this interrupt is acknowledged; that is, until the loader completes.

Copyright © 2012–2019, Texas Instruments Incorporated

6.6.9.4 C-Boot ROM Error Status Returns for MTOCIPC Commands

This section explains the Error or NAK status values that are returned by C-Boot ROM in response to IPC commands shown in Table 6-21.

If C-Boot ROM detects an unsupported IPC command in MTOCIPCCOM register or if IPC flags are not set properly then it will NAK the IPC command by clearing only MTOCIPCFLG[0] and writes an error code onto MTOCIPCDATAR register so that the master subsystem application software will have more information on why the command in NAK'd.

The 'value' column in below table tells the actual value written in MTOCIPCDATAR register bits 15-0, when the command is NAK'd by the C-Boot ROM.

Table 6-22. C-Boot ROM NAK/ERROR Status Returns for MTOCIPCCOM

Value	MTOCIPCDATAR[15:0]	Description
0	CONTROL_SYSTEM_NAK_STATUS_INVALID_VALUE	invalid status - tells corresponding system that control system has not filled in a valid value yet
1	CONTROL_SYSTEM_NAK_STATUS_CMD_NOT_SUPPORTED	tells corresponding system that control system has received a command which is not supported
2	CONTROL_SYSTEM_NAK_STATUS_CMD_NOT_SET_PROPERLY	tells corresponding system that control system has received a command, but the IPCFLG[0] and IPCFLG[31] both are not set.
3	CONTROL_SYSTEM_NAK_STATUS_ALREADY_BUSY_WITH_YOUR_CMD	Tells corresponding system that control system has received a second command from the same system while it is still processing a prev. one.
4	C_BOOTROM_NAK_STATUS_CMD_RESULTED_IN_ERROR	Tells corresponding system that control system tried to execute the command but it resulted in error
5	C_BOOTROM_NAK_STATUS_CMD_CANNOT_BE_EXECUTED_NOW	Tells corresponding system that the control subsystem cannot execute the command now

Whenever an IPC command is NAK'd, then MTOCIPCDATAR[31:16] bits will be set as defined in Section 6.6.10 and MTOCIPCDATAR[15:0] are set as per the above table.

The master subsystem application software can use this feature to find out the health status of C-Boot ROM anytime. Means if master gives an unsupported IPC command to the control subsystem which will be Nak'd and the DATAR register will contain the current health status of C-BootROM.

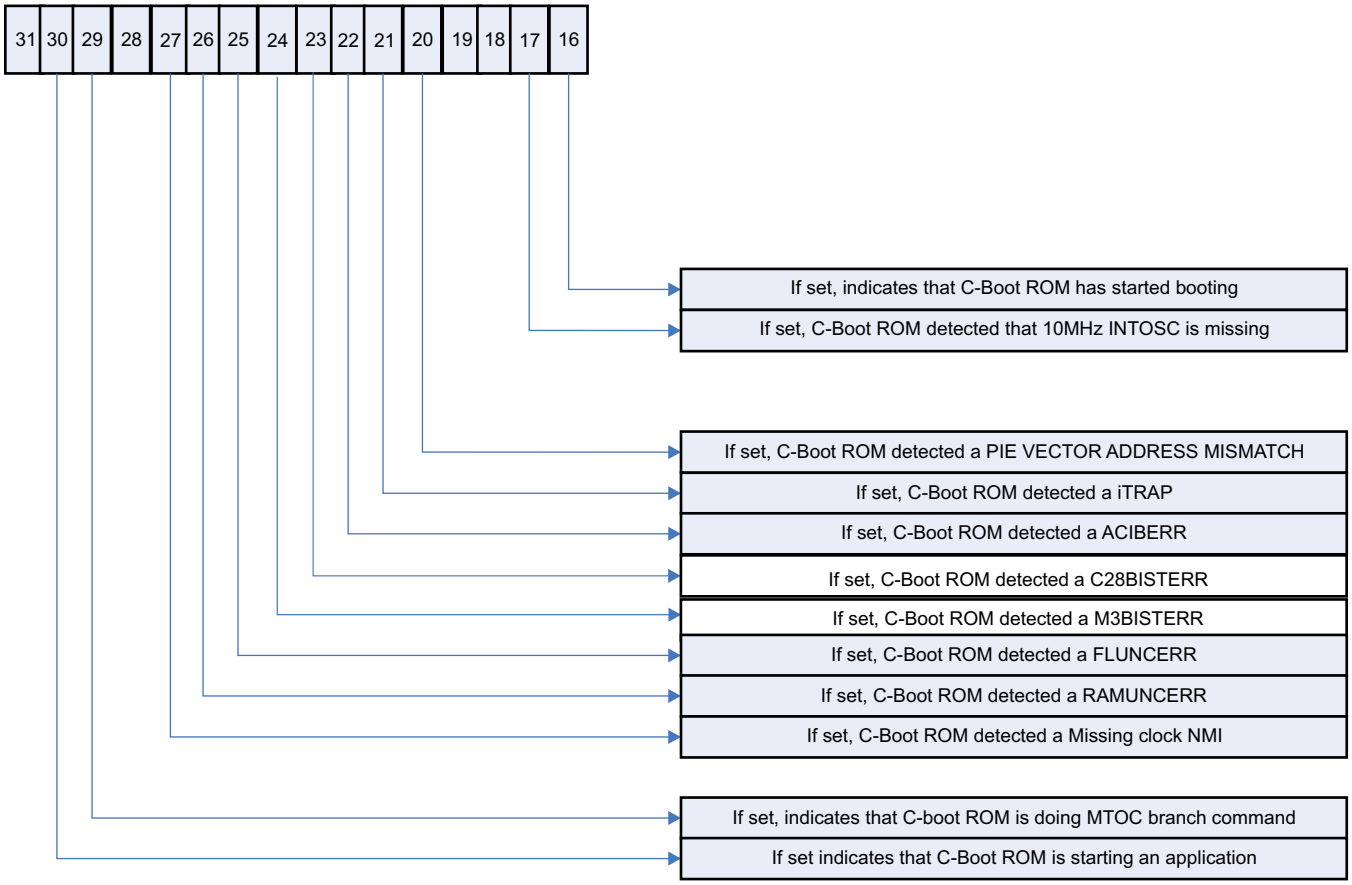
6.6.10 C-Boot ROM Health Status

C-Boot ROM logs its health status onto the CTOMBOOTSTS register bit 31:16 as defined below. The same status in the same format below will be written to MTOCIPCDATAR[31:16] register bits whenever C-Boot ROM Naks an IPC command from the master subsystem.

Note: The rest of the bits are reserved and user applications don't care if these bits are set or reset.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-15. C-Boot ROM Health Status



6.6.11 C-Boot ROM Boot Status

Apart from the health status discussed in Section 6.6.10, C-Boot ROM also logs the boot status in CTOMBOOTSTS register bits 15:0. Table 6-23 shows the same.

Table 6-23. C-Boot ROM Boot Status Values

Value	MTOCIPCBOOTSTS[15:0]	Description
0	C_BOOTROM_BOOTSTS_CTOM_IGNORE	Invalid status - tells corresponding system that control system has not filled in a valid value yet
1	C_BOOTROM_BOOTSTS_CTOM_CONTROL_SYSTEM_START_BOOT	Tells master system that Control system has started to boot, but not completed the boot process yet
2	C_BOOTROM_BOOTSTS_CTOM_CONTROL_SYSTEM_READY	Tells the master system that Control system has completed the boot and is running. It is ready to accept IPC commands from the master.
3	C_BOOTROM_BOOTSTS_CTOM_BOOT_CMD_ACK	Tells the master system that C-BootROM ACKs the command in the MTOCBOOTMODE register
4	C_BOOTROM_BOOTSTS_CTOM_BOOT_CMD_NAK_STATUS_NOT_SUPPORTED	Tells the master system that C-BootROM doesn't support the command in the MTOCBOOTMODE register
5	C_BOOTROM_BOOTSTS_CTOM_BOOT_CMD_NAK_STATUS_BUSY_WITH_BOOT	Tells the master system that C-BootROM NAKs the current boot command in the MTOCBOOTMODE register, as it is already busy with boot.

Master subsystem applications should look for this status before sending any IPC command to C-Boot ROM. For ex: if MTOCIPCBOOTSTS == C_BOOTROM_BOOTSTS_CTOM_CONTROL_SYSTEM_READY, it means that C-Boot ROM is ready to accept MTOC IPC commands.

Copyright © 2012–2019, Texas Instruments Incorporated

6.6.12 C-Boot ROM CTOM IPC Messages

This section explains IPC Messages that will be sent by C-Boot ROM to the master subsystem application. It is up to the application to either handle this IPC commands or ignore these IPC commands from the control subsystem boot ROM. M-Boot ROM ignores these IPC commands from C-Boot ROM.

C-Boot ROM sends IPC messages to the master subsystem to inform of error events that occur asynchronously in the control subsystem.

6.6.12.1 Procedure Followed by C-Boot ROM to Send CTOM IPC Messages

Below is the procedure followed by C-Boot ROM to send IPC messages to Master.

1. CTOMIPCCOM = as described in Table 6-24.
2. CTOMIPCFLG[0] = CTOMIPCFLG[31] = 1; this will trigger CTOMIPCINT1 on the master subsystem, if interrupts are enabled. C-BootROM will not wait for the master subsystem to clear/acknowledge the IPCflags, but the master subsystem software must acknowledge the IPC flags in order to receive next event successfully.

If C-Boot ROM detects that the CTOMIPCFLG bits are not cleared from prev. message it will not send the current IPC message and simply ignore the status.

6.6.12.2 CTOMIPC Messages Table

C-Boot ROM uses CTOMIPC registers to inform the master subsystem if there was an ITRAP exception or if there was a spurious PIE interrupt or if there was a PIE vector address mismatch exception. The events that trigger C-Boot ROM to send these IPC Messages are explained in Section 6.6.12 and Table 6-24.

In this table, the Value column gives the actual data written to CTOMIPCCOM register for the respective IPC message listed in CTOMIPCCOM column. The contents on CTOMIPCADDR, CTOMIPCDATAW and CTOMIPCDATAR columns show the respective register contents when C-Boot ROM is sending the respective message to the master. It is up to the master subsystem application to use these values.

Table 6-24. CTOM IPC Messages

Value	CTOMIPCCOM (M3 - R, B.C28X R/W)	CTOMIPCADDR (M3 - R, B.C28X R/W)	CTOMIPCDATAW (M3 - R, B.C28X R/W)	CTOMIPCDATAR (M3 – R/W, B.C28X-R)	Description
0	CONTROL_IPC_CTOM_COMMAND_IGNORE	DON'T CARE	DON'T CARE	DON'T CARE	Invalid CTOMIPCCOM value, when there is no message sent – default value
0xFFFFFFFE	CONTROL_IPC_CTOM_CONTROL_SYSTEM_IN_ITRAP	Address where an iTRAP occurred	DON'T CARE	DON'T CARE	C-Boot ROM got an iTRAP exception and is waiting for a reset from the master subsystem
0xFFFFFFFD	CONTROL_IPC_CTOM_PIE_INTERRUPT_NOT_SUPPORTED	DON'T CARE	Unsupported PIE interrupt number	DON'T CARE	Tells the master system that C-Boot ROM received an unsupported PIE interrupt, and the interrupt is acknowledged and ignored. C-Boot ROM continues its execution after sending this IPC message
0xFFFFFFFC	CONTROL_IPC_CTOM_CONTROL_SYSTEM_IN_PIE_VECTOR_ADDRESS_MISMATCH	DON'T CARE	DON'T CARE	DON'T CARE	Tells the master system that C-Boot ROM detected PIE vector address mismatch. C-Boot ROM is waiting for a reset from the master subsystem, when this even occurs

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-24. CTOM IPC Messages (continued)

Value	CTOMIPCCOM (M3 - R, B.C28X R/W)	CTOMIPCADDR (M3 - R, B.C28X R/W)	CTOMIPCDATAW (M3 - R, B.C28X R/W)	CTOMIPCDATAR (M3 – R/W, B.C28X-R)	Description
0xFFFFFFFB	C_BOOTROM_IPC_CTOM_CONTROL_SYSTEM_IN_FLUNCERR	DON'T CARE	DON'T CARE	DON'T CARE	Tells the master system that C-Boot ROM detected a Flash uncorrectable error. C-Boot ROM is waiting for a reset from the master subsystem, when this even occurs
0xFFFFFFFA	C_BOOTROM_IPC_CTOM_CONTROL_SYSTEM_IN_RAMUNCERR	DON'T CARE	DON'T CARE	DON'T CARE	Tells the master system that C-Boot ROM detected a RAM uncorrectable error C-Boot ROM is waiting for a reset from the master subsystem, when this even occurs

NOTE: The master subsystem application software should clear CTOMIPCFLG[0] and CTOMIPCFLG[31] bits as soon as it receives the respective messages in order to 'not' miss another IPC status message from C-Boot ROM.

6.6.13 C-Boot ROM Handling of Exceptions and PIE Interrupts

Table 6-25 explains the actions taken by C-Boot ROM in response to various events that can occur during boot.

Table 6-25. C-Boot ROM Exceptions Handling

Exception Event Source	Description	C-Boot ROM action	C-Boot ROM state after exception
CLOCKFAIL – from missing clock detection logic	CLKFAIL condition detected	cbrom_handle_nmi :-> Clear NMI Flags, Save error status in CTOMBOOTSTS register bits and returns from the interrupt handler. NMI is generated to the master subsystem also, No IPC message is sent.	Continue to boot, because missing clock circuit will switch CPU to 10 MHz clock source
M3BISTERR	M3 HW BIST Error NMI Flag	Default NMI HANDLER:-> Clear NMI Flags, Save error status in CTOMBOOTSTS register bits and wait in while(1) loop for master to handle the error state. NMI is generated to C28 also, so no need to send an IPC message.	Wait in While(1) loop, for reset from master.
C28BISTERR	C28 HW BIST Error NMI Flag	Default NMI HANDLER:-> Clear NMI Flags, Save error status in CTOMBOOTSTS register bits and wait in while(1) loop for master to handle the error state. NMI is generated to M3 also, so no need to send an IPC message.	Wait in While(1) loop, for reset from master.
C28RAMUNCERR	C28 RAM Uncorrectable Error NMI Flag	cbrom_handle_nmi :-> Clear NMI Flags, Save error status in CTOMBOOTSTS register, send IPC message to master and wait in while(1) loop for master to handle the error state..	Wait in While(1) loop, for reset frommaster.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-25. C-Boot ROM Exceptions Handling (continued)

Exception Event Source	Description	C-Boot ROM action	C-Boot ROM state after exception
C28FLUNCERR	C28 Flash Uncorrectable Error NMI	cbrom_handle_nmi :-> Clear NMI Flags, Save error status in CTOMBOOTSTS register, send IPC message to master and wait in while(1) loop for master to handle the error state	Wait in While(1) loop, for reset frommaster.
ACIBERR	CIB Error NMI Flag	cbrom_handle_nmi :-> Clear NMI Flags, Save error status in CTOMBOOTSTS register bits and wait in while(1) loop for master to handle the error state. NMI is generated to M3 also, so no need to send an IPC message.	Wait in While(1) loop, for reset from master.
ILLEGAL	ITRAP exception	cbrom_itrap_isr :-> IPC Message is sent to master and iTrap address is written to CTOMIPCADDR. CTOMBOOTSTS register is set to reflect the error status and wait in while(1) loop for master to handle the error state.	Wait in While(1) loop, for reset frommaster.
PIE VECTOR ADDRESS MISMATCH	PIE vector fetch mismatch handler at 0x3FFFBE	Update CTOMBOOTSTS to reflect the error, and the ROM handler @0x3FFFBE will send an IPC message to master and wait in while(1) loop for master to handle the error state.	Wait in While(1) loop, for reset from master.
Spurious PIE interrupts	Any un-supported PIE interrupt occurs	Send an IPC message to master, with the spurious interrupt no. ACK the spurious interrupt and continue doing whatever C-Boot ROM was doing before	Continue to boot

6.6.14 C-Boot ROM Bootloader Functionality

This section explains each of the supported peripheral boot mode(s) on C-Boot ROM, in detail.

On Concerto, the data transfer protocols or stream structures that allow boot data transfer between C-Boot ROM and Host device, are compatible to the respective bootloaders on Piccolo class of C2000 devices. This allows user to re-use the same tools used to send boot data to Piccolo(s) also to send boot data to C-Boot ROM on Concerto devices.

6.6.14.1 C-Boot Data Stream Structure

The following two tables and associated examples show the structure of the data stream incoming to the bootloader. The basic structure is the same for all the bootloaders and is based on the C54x source data stream generated by the C54x hex utility. The C28x hex utility (hex2000.exe) has been updated to support this structure. The hex2000.exe utility is included with the C2000 code generation tools. All values in the data stream structure are in hex.

The first 16-bit word in the data stream is known as the key value. The key value is used to tell the bootloader the width of the incoming stream: 8 or 16 bits. Note that not all bootloaders will accept both 8 and 16-bit streams. Please refer to the detailed information on each loader for the valid data stream width. For an 8-bit data stream, the key value is 0x08AA and for a 16-bit stream it is 0x10AA. If a bootloader receives an invalid key value, then the load is aborted.

Copyright © 2012–2019, Texas Instruments Incorporated

The next eight words are used to initialize register values or otherwise enhance the bootloader by passing values to it. If a bootloader does not use these values then they are reserved for future use and the bootloader simply reads the value and then discards it. Currently only the SPI and I2C and parallel XINTF bootloaders use these words to initialize registers.

The tenth and eleventh words comprise the 22-bit entry point address. This address is used to initialize the PC after the boot load is complete. This address is most likely the entry point of the program downloaded by the bootloader.

The twelfth word in the data stream is the size of the first data block to be transferred. The size of the block is defined for both 8-bit and 16-bit data stream formats as the number of 16-bit words in the block. For example, to transfer a block of 20 8-bit data values from an 8-bit data stream, the block size would be 0x000A to indicate 10 16-bit words.

The next two words tell the loader the destination address of the block of data. Following the size and address will be the 16-bit words that make up that block of data.

This pattern of block size/destination address repeats for each block of data to be transferred. Once all the blocks have been transferred, a block size of 0x0000 signals to the loader that the transfer is complete. At this point the loader will return the entry point address to the calling routine which in turn will cleanup and exit. Execution will then continue at the entry point address as determined by the input data stream contents.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-26. General Structure Of Source Program Data Stream In 16-Bit Mode

Word	Contents
1	10AA (KeyValue for memory width = 16bits)
2	Register initialization value or reserved for future use
3	Register initialization value or reserved for future use
4	Register initialization value or reserved for future use
5	Register initialization value or reserved for future use
6	Register initialization value or reserved for future use
7	Register initialization value or reserved for future use
8	Register initialization value or reserved for future use
9	Register initialization value or reserved for future use
10	Entry point PC[22:16]
11	Entry point PC[15:0]
12	Block size (number of words) of the first block of data to load. If the block size is 0, this indicates the end of the source program. Otherwise another section follows.
13	Destination address of first block Addr[31:16]
14	Destination address of first block Addr[15:0]
15	First word of the first block in the source being loaded
...	...
...	...
.	Last word of the first block of the source being loaded
.	Block size of the 2nd block to load.
.	Destination address of second block Addr[31:16]
.	Destination address of second block Addr[15:0]
.	First word of the second block in the source being loaded
.	…
.	Last word of the second block of the source being loaded
.	Block size of the last block to load
.	Destination address of last block Addr[31:16]
.	Destination address of last block Addr[15:0]
.	First word of the last block in the source being loaded
...	...
...	...
n	Last word of the last block of the source being loaded
n+1	Block size of 0000h - indicates end of the source program

Copyright © 2012–2019, Texas Instruments Incorporated

Example 6-3. Data Stream Structure 16-bit

```
10AA                ; 0x10AA 16-bit key value
0000 0000 0000 0000 ; 8 reserved words
0000 0000 0000 0000
003F 8000           ; 0x003F8000 EntryAddr, starting point after boot load completes
0005                ; 0x0005 - First block consists of 5 16-bit words
003F 9010           ; 0x003F9010 - First block will be loaded starting at 0x3F9010
0001 0002 0003 0004 ; Data loaded = 0x0001 0x0002 0x0003 0x0004 0x0005
0005
0002                ; 0x0002 - 2nd block consists of 2 16-bit words
003F 8000           ; 0x003F8000 - 2nd block will be loaded starting at 0x3F8000
7700 7625           ; Data loaded = 0x7700 0x7625
0000                ; 0x0000 - Size of 0 indicates end of data stream
After load has completed the following memory values will have been initialized as follows:
Location     Value
0x3F9010     0x0001
0x3F9011     0x0002
0x3F9012     0x0003
0x3F9013     0x0004
0x3F9014     0x0005
0x3F8000     0x7700
0x3F8001     0x7625
PC Begins execution at 0x3F8000
```

In 8-bit mode, the least significant byte (LSB) of the word is sent first followed by the most significant byte (MSB). For 32-bit values, such as a destination address, the most significant word (MSW) is loaded first, followed by the least significant word (LSW). The bootloaders take this into account when loading an 8-bit data stream.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-27. LSB/MSB Loading Sequence in 8-Bit Data Stream

Byte		Contents	
		LSB (First Byte of 2)	MSB (Second Byte of 2)
1	2	LSB: AA (KeyValue for memory width = 8 bits)	MSB: 08h (KeyValue for memory width = 8 bits)
3	4	LSB: Register initialization value or reserved	MSB: Register initialization value or reserved
5	6	LSB: Register initialization value or reserved	MSB: Register initialization value or reserved
7	8	LSB: Register initialization value or reserved	MSB: Register initialization value or reserved
...	...	...	...
...	...	...	...
17	18	LSB: Register initialization value or reserved	MSB: Register initialization value or reserved
19	20	LSB: Upper half of Entry point PC[23:16]	MSB: Upper half of entry point PC[31:24] (Always 0x00)
21	22	LSB: Lower half of Entry point PC[7:0]	MSB: Lower half of Entry point PC[15:8]
23	24	LSB: Block size in words of the first block to load. If the block size is 0, this indicates the end of the source program. Otherwise another block follows. For example, a block size of 0x000A would indicate 10 words or 20 bytes in the block.	MSB: block size
25	26	LSB: MSW destination address, first block Addr[23:16]	MSB: MSW destination address, first block Addr[31:24]
27	28	LSB: LSW destination address, first block Addr[7:0]	MSB: LSW destination address, first block Addr[15:8]
29	30	LSB: First word of the first block being loaded	MSB: First word of the first block being loaded
...	...	...	...
...	...	...	...
.	.	LSB: Last word of the first block to load	MSB: Last word of the first block to load
.	.	LSB: Block size of the second block	MSB: Block size of the second block
.	.	LSB: MSW destination address, second block Addr[23:16]	MSB: MSW destination address, second block Addr[31:24]
.	.	LSB: LSW destination address, second block Addr[7:0]	MSB: LSW destination address, second block Addr[15:8]
.	.	LSB: First word of the second block being loaded	MSB: First word of the second block being loaded
...	...	...	...
...	...	...	...
.	.	LSB: Last word of the second block	MSB: Last word of the second block
.	.	LSB: Block size of the last block	MSB: Block size of the last block
.	.	LSB: MSW of destination address of last block Addr[23:16]	MSB: MSW destination address, last block Addr[31:24]
.	.	LSB: LSW destination address, last block Addr[7:0]	MSB: LSW destination address, last block Addr[15:8]
.	.	LSB: First word of the last block being loaded	MSB: First word of the last block being loaded
...	...	...	...
...	...	...	...
.	.	LSB: Last word of the last block	MSB: Last word of the last block
n	n+1	LSB: 00h	MSB: 00h - indicates the end of the source

Copyright © 2012–2019, Texas Instruments Incorporated

Example 6-4. Data Stream Structure 8-bit

```
AA 08          ; 0x08AA 8-bit key value
00 00 00 00    ; 8 reserved words
00 00 00 00
00 00 00 00
00 00 00 00
3F 00 00 80    ; 0x003F8000 EntryAddr, starting point after boot load completes
05 00          ; 0x0005 - First block consists of 5 16-bit words
3F 00 10 90    ; 0x003F9010 - First block will be loaded starting at 0x3F9010
01 00          ; Data loaded = 0x0001 0x0002 0x0003 0x0004 0x0005
02 00
03 00
04 00
05 00
02 00          ; 0x0002 - 2nd block consists of 2 16-bit words
3F 00 00 80    ; 0x003F8000 - 2nd block will be loaded starting at 0x3F8000
00 77          ; Data loaded = 0x7700 0x7625
25 76
00 00          ; 0x0000 - Size of 0 indicates end of data stream

After load has completed the following memory values will have been initialized as follows:

Location   Value
0x3F9010   0x0001
0x3F9011   0x0002
0x3F9012   0x0003
0x3F9013   0x0004
0x3F9014   0x0005
0x3F8000   0x7700
0x3F8001   0x7625
PC Begins execution at 0x3F8000
```

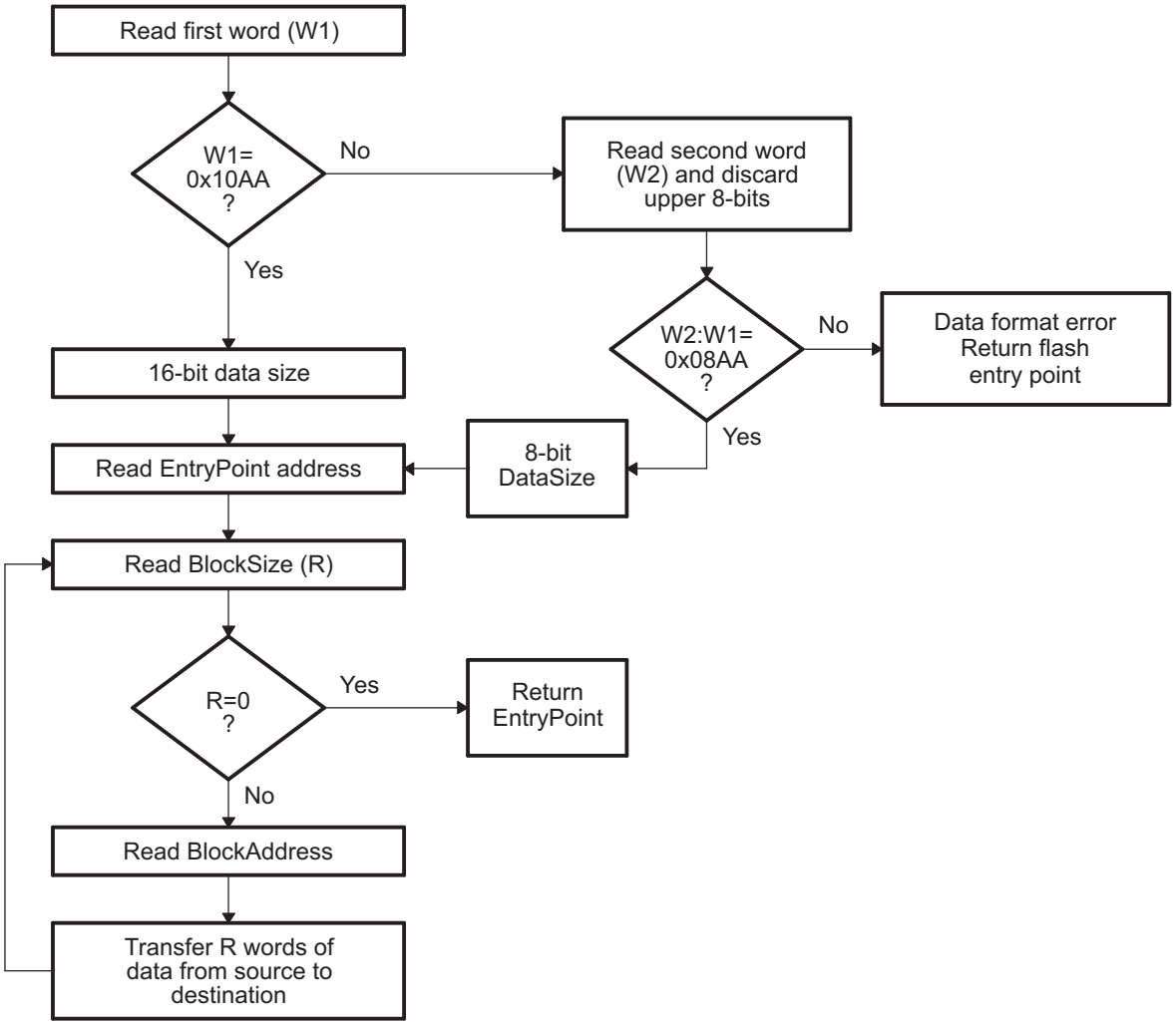
6.6.14.2 Basic Data Transfer Procedure

Figure 6-16 illustrates the basic process a bootloader uses to determine whether 8-bit or 16-bit data stream has been selected, transfer that data, and start program execution. This process occurs after the bootloader finds the valid boot mode selected by the state of $\overline{\text{TRST}}$ and GPIO pins.

The loader first compares the first value sent by the host against the 16-bit key value of 0x10AA. If the value fetched does not match then the loader will read a second value. This value will be combined with the first value to form a word. This will then be checked against the 8-bit key value of 0x08AA. If the loader finds that the header does not match either the 8-bit or 16-bit key value, or if the value is not valid for the given boot mode then the load will abort.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-16. Bootloader Basic Transfer Procedure



8-bit and 16-bit transfers are not valid for all boot modes. If only one mode is valid, then this decision tree is skipped and the key value is only checked for correctness. See the info specific to a particular bootloader for any limitations.

In 8-bit mode, the LSB of the 16-bit word is read first followed by the MSB.

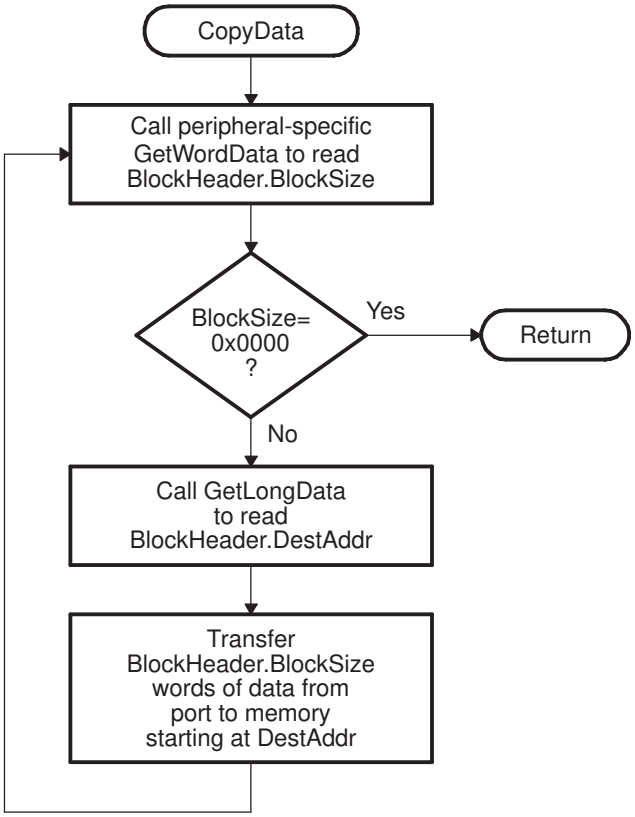
Copyright © 2012–2019, Texas Instruments Incorporated

6.6.14.3 CopyData Function

Each of the bootloaders uses the same function to copy data from the port to the device's SARAM. This function is the CopyData() function. This function uses a pointer to a GetWordData function that is initialized by each of the loaders to properly read data from that port. For example, when the SPI loader is evoked, the GetWordData function pointer is initialized to point to the SPI-specific SPI_GetWordData function. Thus when the CopyData() function is called, the correct port is accessed. The flow of the CopyData function is shown in Figure 6-17.

Note: BlockSize must be less than 0xFFFF for correct operation of the CopyData function. This means the max possible value of BlockSize is 0xFFFE, not 0xFFFF.

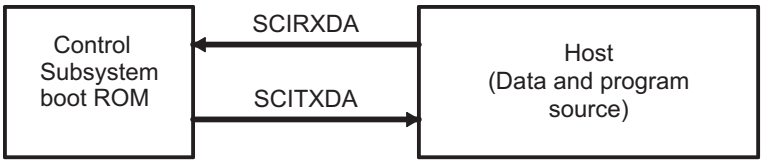
Figure 6-17. Overview of CopyData Function



6.6.14.4 C-Boot ROM SCI_Boot Mode

The SCI boot mode asynchronously transfers code from SCI-A to internal memory. This boot mode only supports an incoming 8-bit data stream and follows the same data flow as outlined in Example 6-4.

Figure 6-18. Overview of SCI Bootloader Operation



The SCI-A loader uses following pins:

- SCIRXDA on GPIO36
- SCITXDA on GPIO35

Copyright © 2012–2019, Texas Instruments Incorporated

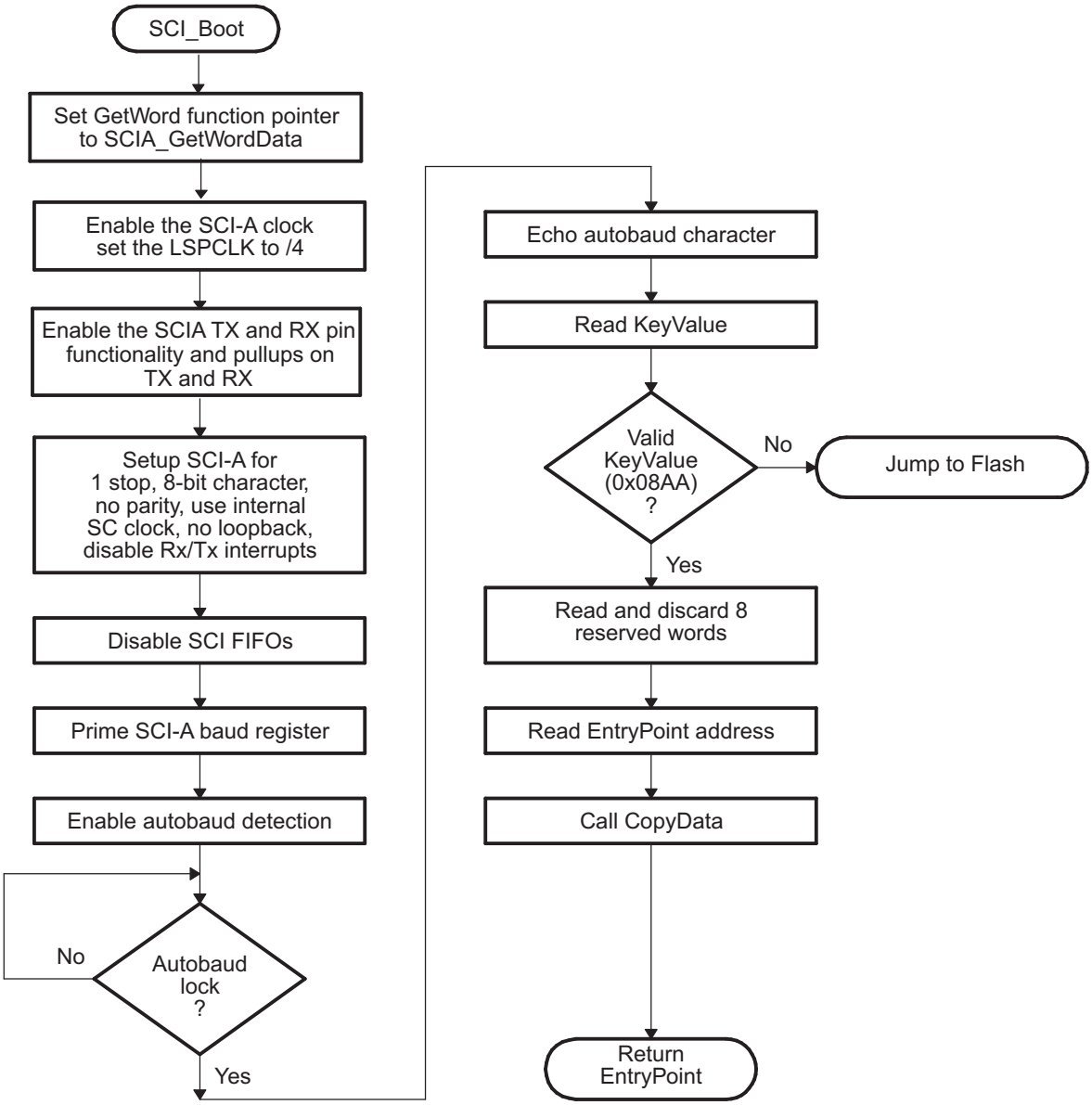
The device communicates with the external host device by communication through the SCI-A peripheral. The autobaud feature of the SCI port is used to lock baud rates with the host. For this reason the SCI loader is very flexible and you can use a number of different baud rates to communicate with the device.

After each data transfer, the 28x will echo back the 8-bit character received to the host. In this manner, the host can perform checks that each character was received by the 28x.

At higher baud rates, the slew rate of the incoming data bits can be effected by transceiver and connector performance. While normal serial communications may work well, this slew rate may limit reliable auto-baud detection at higher baud rates (typically beyond 100kbaud) and cause the auto-baud lock feature to fail. To avoid this, the following is recommended:

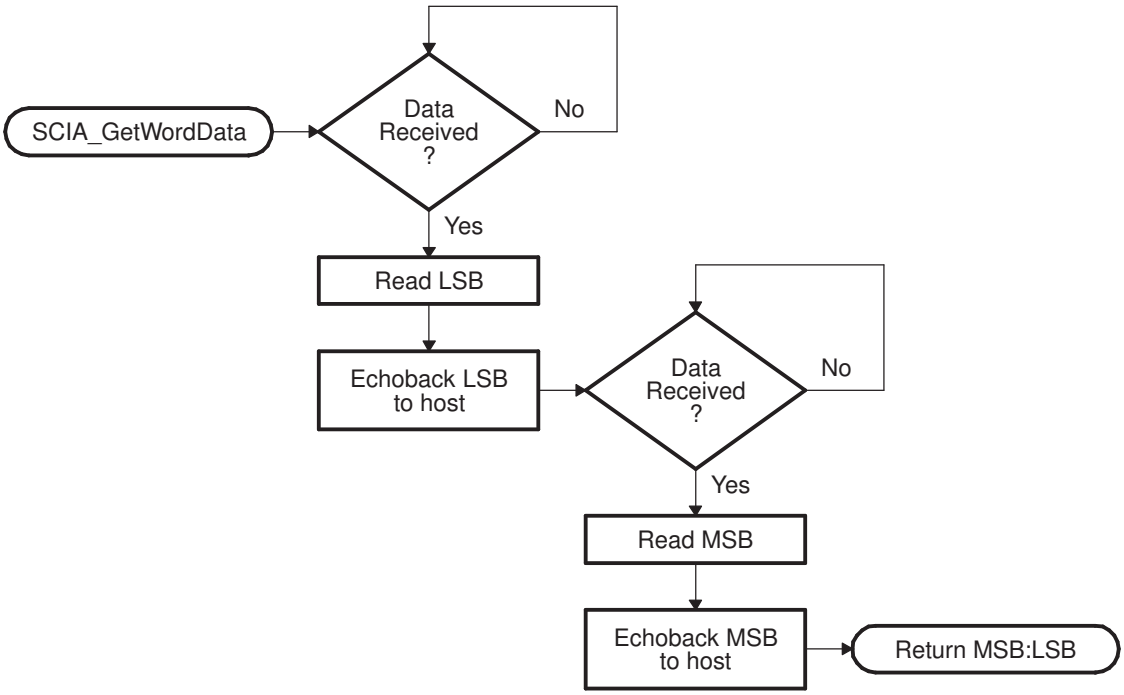
1. Achieve a baud-lock between the host and 28x SCI bootloader using a lower baud rate.
2. Load the incoming 28x application or custom loader at this lower baud rate.
3. The host may then handshake with the loaded 28x application to set the SCI baud rate register to the desired high baud rate.

Figure 6-19. Overview of SCI_Boot Function



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-20. Overview of SCI_GetWordData Function



Copyright © 2012–2019, Texas Instruments Incorporated

6.6.14.5 C-Boot ROM SPI_Boot Mode

The SPI loader expects an SPI-compatible 16-bit or 24-bit addressable serial EEPROM or serial flash device to be present on the SPI-A pins as indicated in Figure 6-21. The SPI bootloader supports an 8-bit data stream. It does not support a 16-bit data stream.

Figure 6-21. SPI Loader

The SPI-A loader uses following pins:

- SPISIMOA on GPIO16
- SPISOMIA on GPIO17
- SPICLKA on GPIO18
- SPISTEA on GPIO19

The SPI boot ROM loader initializes the SPI module to interface to a serial SPI EEPROM or flash. Devices of this type include, but are not limited to, the Xicor X25320 (4Kx8) and Xicor X25256 (32Kx8) SPI serial SPI EEPROMs and the Atmel AT25F1024A serial flash.

The SPI boot ROM loader initializes the SPI with the following settings: FIFO enabled, 8-bit character, internal SPICLK master mode and talk mode, clock phase = 1, polarity = 0, using the slowest baud rate.

If the download is to be performed from an SPI port on another device, then that device must be setup to operate in the slave mode and mimic a serial SPI EEPROM. Immediately after entering the SPI_Boot function, the pin functions for the SPI pins are set to primary and the SPI is initialized. The initialization is done at the slowest speed possible. Once the SPI is initialized and the key value read, you could specify a change in baud rate or low speed peripheral clock.

Table 6-28. SPI 8-Bit Data Stream

Byte	Contents
1	LSB: AA (KeyValue for memory width = 8-bits)
2	MSB: 08h (KeyValue for memory width = 8-bits)
3	LSB: LOSPCP
4	MSB: SPIBRR
5	LSB: reserved for future use
6	MSB: reserved for future use
...	...
...	Data for this section.
	...
17	LSB: reserved for future use
18	MSB: reserved for future use
19	LSB: Upper half (MSW) of Entry point PC[23:16]
20	MSB: Upper half (MSW) of Entry point PC[31:24] (Note: Always 0x00)
21	LSB: Lower half (LSW) of Entry point PC[7:0]
22	MSB: Lower half (LSW) of Entry point PC[15:8]
...	....
...	Data for this section.
	...
...	Blocks of data in the format size/destination address/data as shown in the generic data stream description

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-28. SPI 8-Bit Data Stream (continued)

Byte	Contents
...	...
...	Data for this section.
	...
n	LSB: 00h
n+1	MSB: 00h - indicates the end of the source

The data transfer is done in "burst" mode from the serial SPI EEPROM. The transfer is carried out entirely in byte mode (SPI at 8 bits/character). A step-by-step description of the sequence follows:

Step 1. The SPI-A port is initialized

Step 2. The GPIO19 (SPISTE) pin is used as a chip-select for the serial SPI EEPROM or flash

Step 3. The SPI-A outputs a read command for the serial SPI EEPROM or flash

Step 4. The SPI-A sends the serial SPI EEPROM an address 0x0000; that is, the host requires that the EEPROM or flash must have the downloadable packet starting at address 0x0000 in the EEPROM or flash. The loader is compatible with both 16-bit addresses and 24-bit addresses.

Step 5. The next word fetched must match the key value for an 8-bit data stream (0x08AA). The least significant byte of this word is the byte read first and the most significant byte is the next byte fetched. This is true of all word transfers on the SPI. If the key value does not match, then the load is aborted and

Step 6. The next 2 bytes fetched can be used to change the value of the low speed peripheral clock register (LOSPCP) and the SPI baud rate register (SPIBRR). The first byte read is the LOSPCP value and the second byte read is the SPIBRR value. The next 7 words are reserved for future enhancements. The SPI bootloader reads these 7 words and discards them.

Step 7. The next 2 words makeup the 32-bit entry point address where execution will continue after the boot load process is complete. This is typically the entry point for the program being downloaded through the SPI port.

Step 8. Multiple blocks of code and data are then copied into memory from the external serial SPI EEPROM through the SPI port. The blocks of code are organized in the standard data stream structure presented earlier. This is done until a block size of 0x0000 is encountered. At that point in time the entry point address is returned to the calling routine that then exits the bootloader and resumes execution at the address specified.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-22. Data Transfer From EEPROM Flow

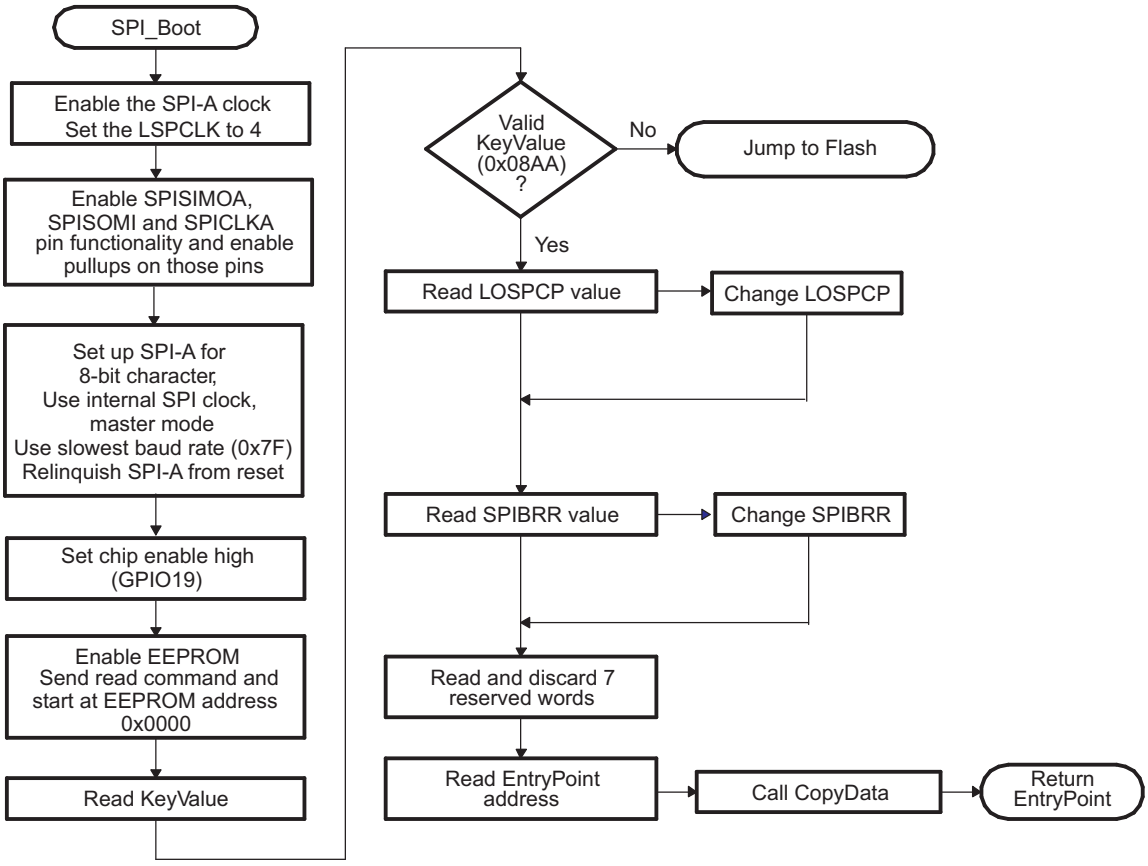
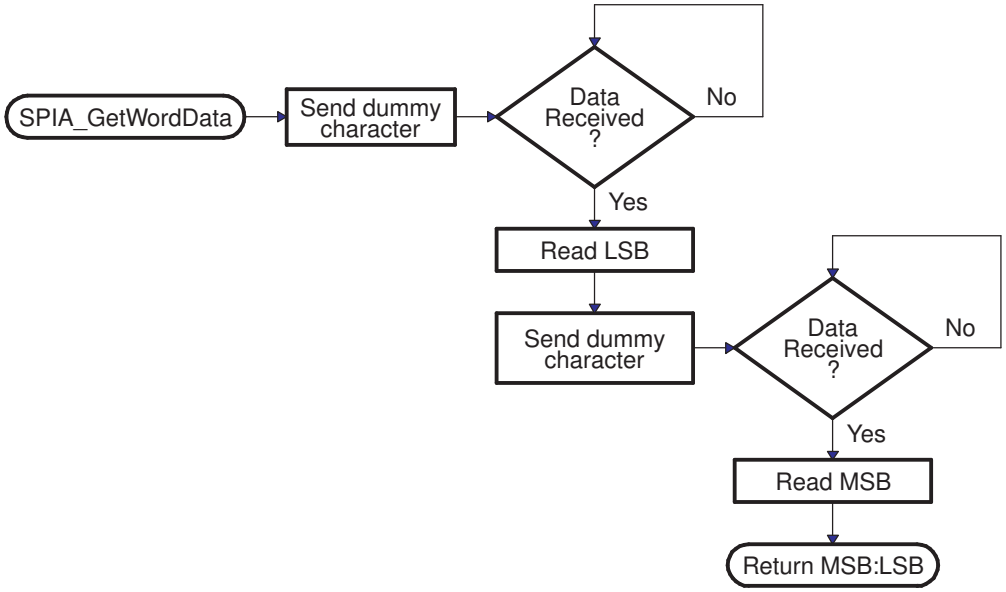


Figure 6-23. Overview of SPIA_GetWordData Function

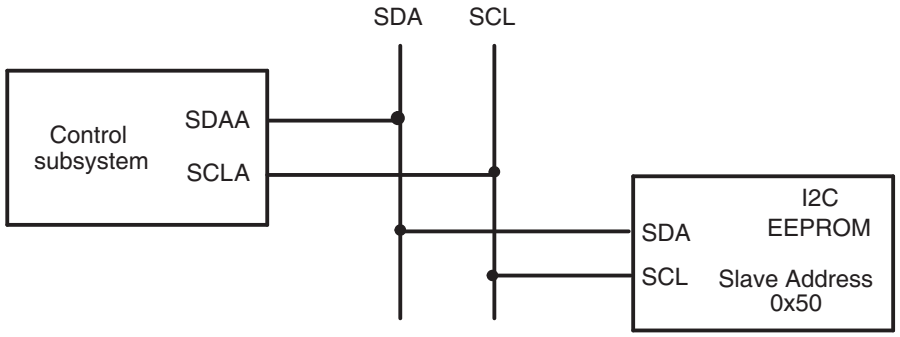


Copyright © 2012–2019, Texas Instruments Incorporated

6.6.14.6 C-Boot ROM I2C Boot Mode

The I2C bootloader expects an 8-bit wide I2C-compatible EEPROM device to be present at address 0x50 on the I2C-A bus as indicated in Figure 6-24. The EEPROM must adhere to conventional I2C EEPROM protocol, as described in this section, with a 16-bit base address architecture.

Figure 6-24. EEPROM Device at Address 0x50



The I2C loader uses following pins:

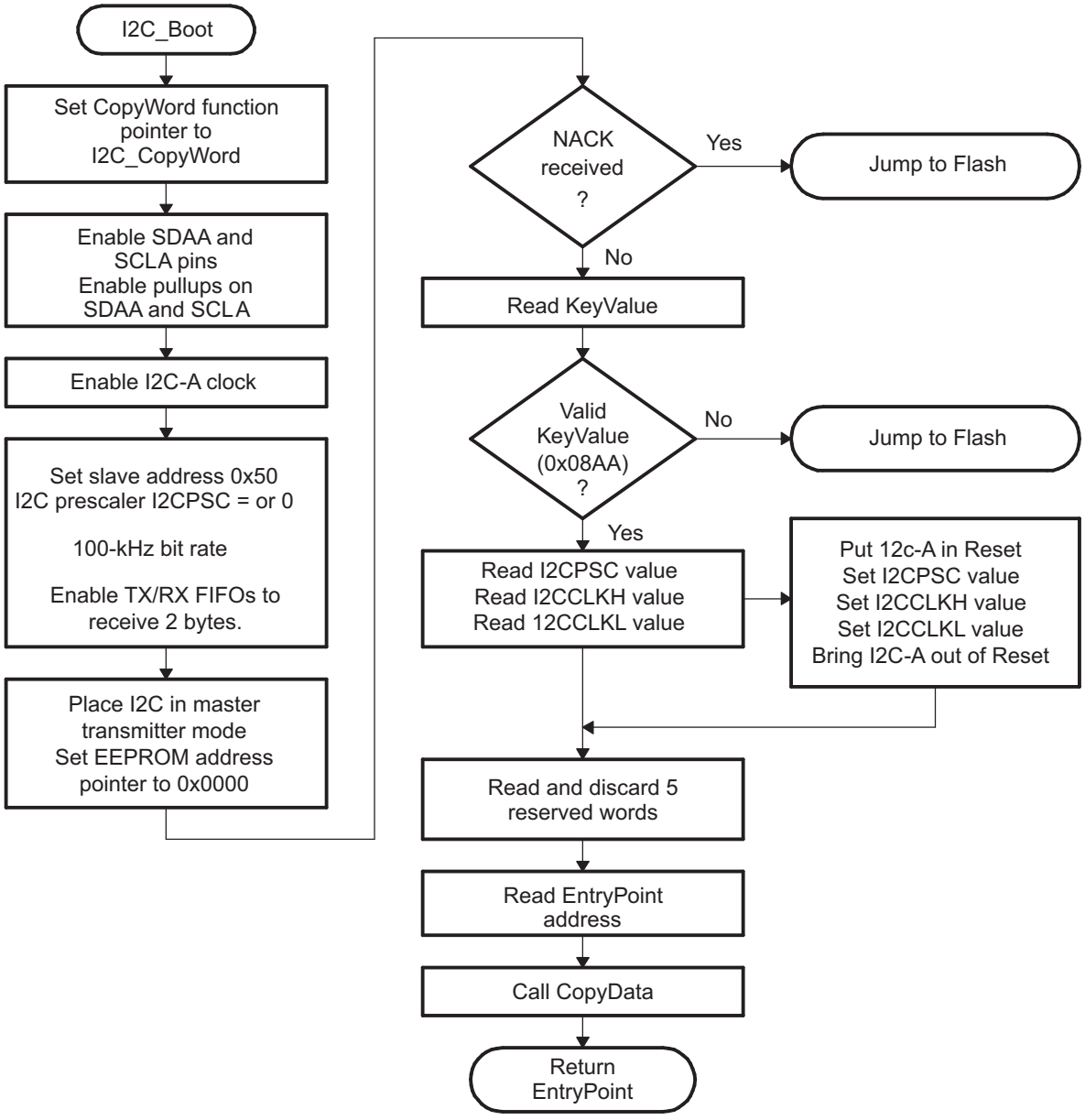
- SDAA on GPIO 32
- SCLA on GPIO 33

If the download is to be performed from a device other than an EEPROM, then that device must be set up to operate in the slave mode and mimic the I2C EEPROM. Immediately after entering the I2C boot function, the GPIO pins are configured for I2C-A operation and the I2C is initialized. The following requirements must be met when booting from the I2C module:

- The input frequency to the device must be in the appropriate range.
- The EEPROM must be at slave address 0x50.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-25. Overview of I2C_Boot Function



The bit-period prescalers (I2CCLKH and I2CCLKL) are configured by the bootloader to run the I2C at a 50 percent duty cycle at 100-kHz bit rate (standard I2C mode) when the system clock is 10 MHz. These registers can be modified after receiving the first few bytes from the EEPROM. This allows the communication to be increased up to a 400-kHz bit rate (fast I2C mode) during the remaining data reads.

Arbitration, bus busy, and slave signals are not checked. Therefore, no other master is allowed to control the bus during this initialization phase. If the application requires another master during I2C boot mode, that master must be configured to hold off sending any I2C messages until the application software signals that it is past the bootloader portion of initialization.

The non-acknowledgment bit is checked only during the first message sent to initialize the EEPROM base address. This is to make sure that an EEPROM is present at address 0x50 before continuing. If an EEPROM is not present, code will The non-acknowledgment bit is not checked during the address phase of the data read messages (I2C_Get Word). If a non acknowledgment is received during the data read messages, the I2C bus will hang. Table 7-3 shows the 8-bit data stream used by the I2C.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-29. I2C 8-Bit Data Stream

Byte		Contents
	1	LSB: AA (KeyValue for memory width = 8 bits)
	2	MSB: 08h (KeyValue for memory width = 8 bits)
	3	LSB: I2CPSC[7:0]
	4	reserved
	5	LSB: I2CCLKH[7:0]
	6	MSB: I2CCLKH[15:8]
	7	LSB: I2CCLKL[7:0]
	8	MSB: I2CCLKL[15:8]
	...	...
	...	Data for this section.
		...
	17	LSB: Reserved for future use
	18	MSB: Reserved for future use
	19	LSB: Upper half of entry point PC
	20	MSB: Upper half of entry point PC[22:16] (Note: Always 0x00)
	21	LSB: Lower half of entry point PC[15:8]
	22	MSB: Lower half of entry point PC[7:0]
	...	...
	...	Data for this section.
		...
		Blocks of data in the format size/destination address/data as shown in the generic data stream description.
	...	...
	...	Data for this section.
		...
LSB: 00h	n	
	n+1	MSB: 00h - indicates the end of the source

The I2C EEPROM protocol required by the I2C bootloader is shown in Figure 6-26 and Figure 6-27. The first communication, which sets the EEPROM address pointer to 0x0000 and reads the KeyValue (0x08AA) from it, is shown in Figure 6-26. All subsequent reads are shown in Figure 6-27 and are read two bytes at a time.

Figure 6-26. Random Read

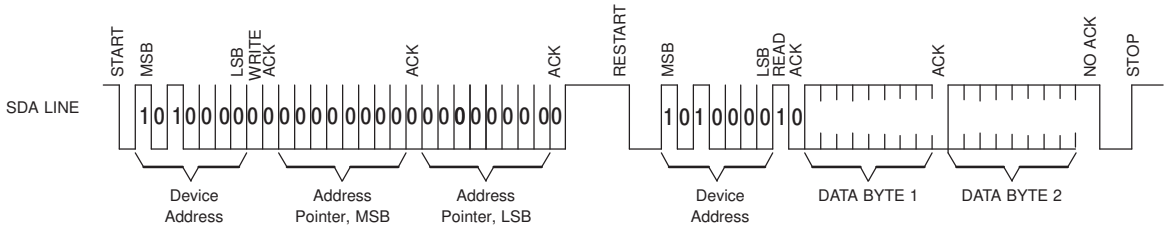
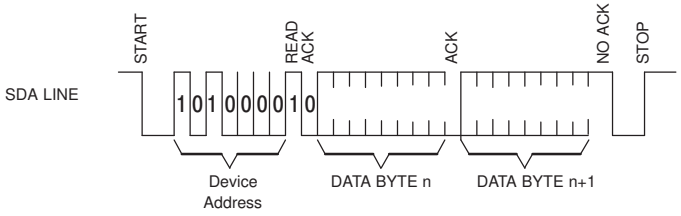


Figure 6-27. Sequential Read

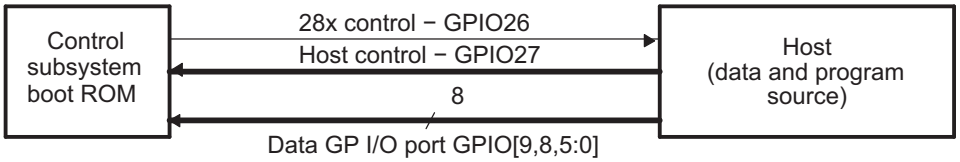


Copyright © 2012–2019, Texas Instruments Incorporated

6.6.14.7 C-Boot ROM Parallel Boot Mode Protocol

The parallel general purpose I/O (GPIO) boot mode asynchronously transfers code from GPIO0 -GPIO5, GPIO8-GPIO9 to internal memory. Each value is 8 bits long and follows the same data flow as outlined in Figure 6-28.

Figure 6-28. Overview of Parallel GPIO Bootloader Operation



The control subsystem communicates with the external host device by polling/driving the GPIO27 and GPIO26 lines. The handshake protocol shown in Figure 6-29 must be used to successfully transfer each word via GPIO [9,8,5:0]. This protocol is very robust and allows for a slower or faster host to communicate with the master subsystem.

Two consecutive 8-bit words are read to form a single 16-bit word. The most significant byte (MSB) is read first followed by the least significant byte (LSB). In this case, data is read from GPIO[9,8,5:0].

The 8-bit data stream is shown in Table 6-30.

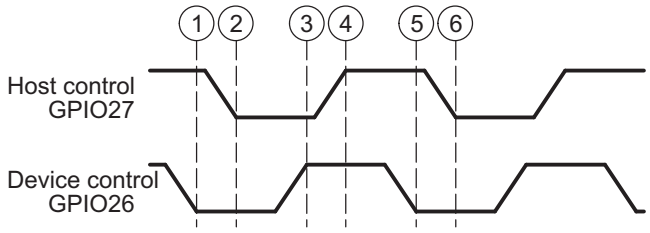
Copyright © 2012–2019, Texas Instruments Incorporated

Table 6-30. Parallel GPIO Boot 8-Bit Data Stream

Bytes		GPIO[9,8,5:0] (Byte 1 of 2)	GPIO[9,8,5:0] (Byte 2 of 2)	Description
1	2	AA	08	0x08AA (KeyValue for memory width = 16bits)
3	4	00	00	8 reserved words (words 2 - 9)
...	...	...	...	...
17	18	00	00	Last reserved word
19	20	BB	00	Entry point PC[22:16]
21	22	DD	CC	Entry point PC[15:0] (PC = 0x00BBCCDD)
23	24	NN	MM	Block size of the first block of data to load = 0xMMNN words
25	26	BB	AA	Destination address of first block Addr[31:16]
27	28	DD	CC	Destination address of first block Addr[15:0] (Addr = 0xAABBCCDD)
29	30	BB	AA	First word of the first block in the source being loaded = 0xAABB
...				...
...				Data for this section.
				...
.		BB	AA	Last word of the first block of the source being loaded = 0xAABB
.		NN	MM	Block size of the 2nd block to load = 0xMMNN words
.		BB	AA	Destination address of second block Addr[31:16]
.		DD	CC	Destination address of second block Addr[15:0]
.		BB	AA	First word of the second block in the source being loaded
.				…
n	n+1	BB	AA	Last word of the last block of the source being loaded (More sections if required)
n+2	n+3	00	00	Block size of 0000h - indicates end of the source program

The device first signals the host that it is ready to begin data transfer by pulling the GPIO26 pin low. The host load then initiates the data transfer by pulling the GPIO27 pin low. The complete protocol is shown in the diagram below:

Figure 6-29. Parallel GPIO Bootloader Handshake Protocol



1. The device indicates it is ready to start receiving data by pulling the GPI2O6 pin low.
2. The bootloader waits until the host puts data on GPIO [9,8,5:0]. The host signals to the device that data is ready by pulling the GPIO27 pin low.
3. The device reads the data and signals the host that the read is complete by pulling GPIO26 high.
4. The bootloader waits until the host acknowledges the device by pulling GPIO27 high.
5. The device again indicates it is ready for more data by pulling the GPIO26 pin low.

This process is repeated for each data value to be sent.

Figure 6-30 shows an overview of the Parallel GPIO bootloader flow.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-30. Parallel GPIO Mode Overview

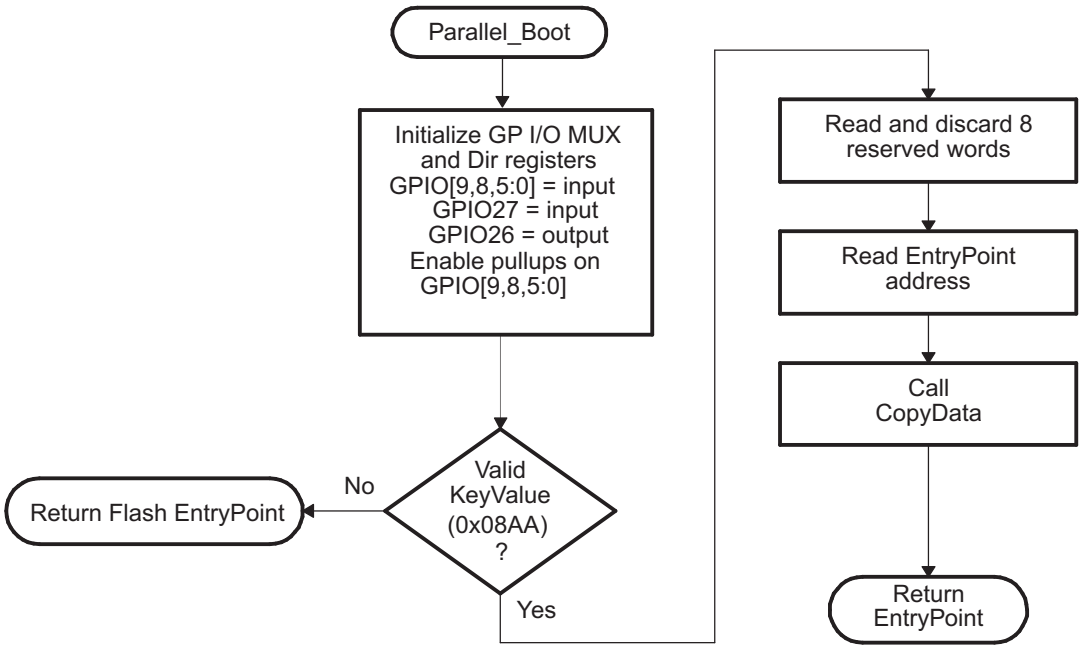


Figure 6-31 shows the transfer flow from the host side. The operating speed of the CPU and host are not critical in this mode as the host will wait for the device and the device will in turn wait for the host. In this manner the protocol will work with both a host running faster and a host running slower than the device.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-31. Parallel GPIO Mode - Host Transfer Flow

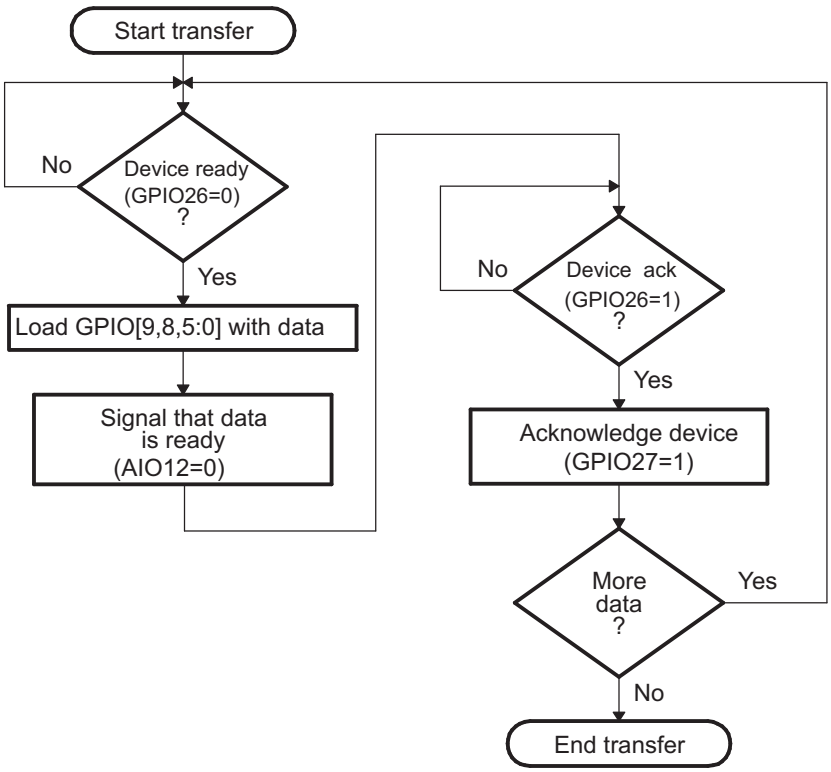


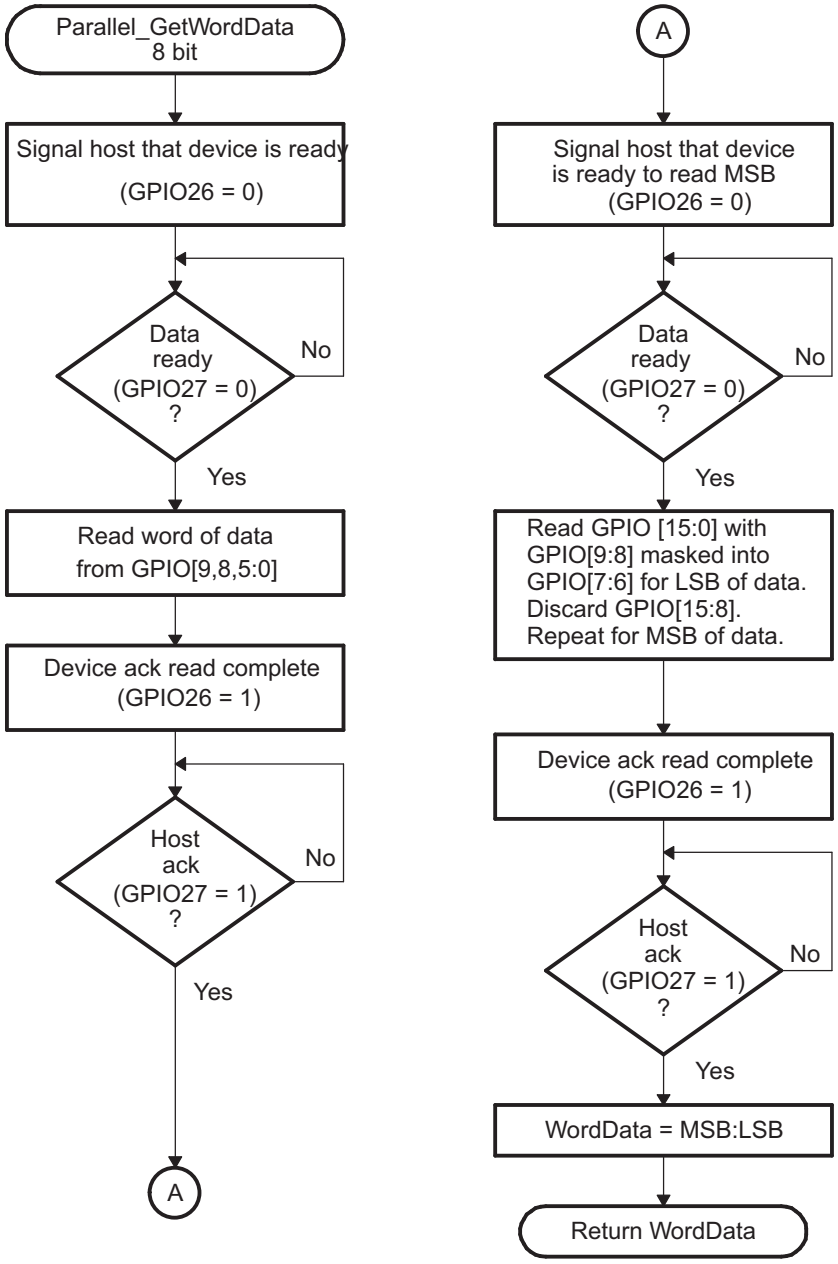
Figure 6-32 shows the flow used to read a single word of data from the parallel port.

- **8-bit data stream**

  The 8-bit routine, shown in Figure 6-32, discards the upper 8 bits of the first read from the port and treats the lower 8 bits masked with GPIO9 in bit position 7 and GPIO8 in bit position 6 as the the least significant byte (LSB) of the word to be fetched. The routine will then perform a second read to fetch the most significant byte (MSB). The routine will then perform a second read to fetch the most significant byte (MSB). It then combines the MSB and LSB into a single 16-bit value to be passed back to the calling routine.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-32. 8-Bit Parallel GetWord Function



6.7 Guidelines for Boot ROM Application Writers

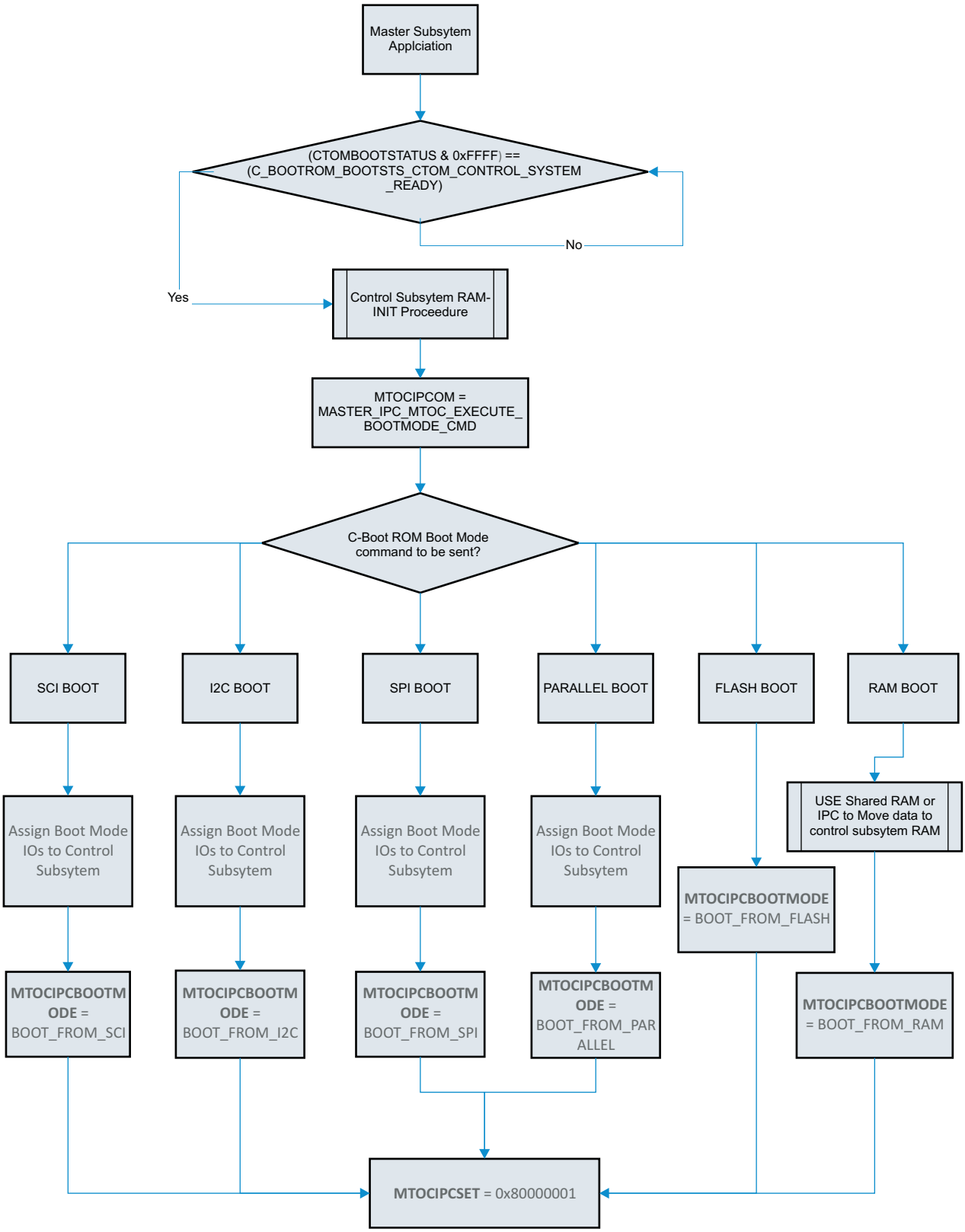
This section provides guidelines to write a master subsystem application in order to kick-start a peripheral bootloader on C-Boot ROM or to let the control subsystem boot to flash.

6.7.1 Master Subsystem Application Procedure to Start C-Boot ROM Bootloaders

As previously explained, M-Boot ROM brings the control subsystem out of reset so that it can execute C-Boot ROM and M-Boot ROM will continue to execute its own application. Mean-while C-Boot ROM initializes the control subsystem as explained and puts C28x CPU in IDLE Low Power Mode. It is up to the master subsystem application to decide on how it wants to let the control subsystem boot and function. The flow diagrams and procedures listed below in this section explain the minimum things that should be done in order to successfully boot the control subsystem.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-33. Master Subsystem Application Flow to Start C-Boot ROM Loaders



Copyright © 2012–2019, Texas Instruments Incorporated

Note: Refer to the next section for more details on the control subsystem RAM-INIT procedure shown in Figure 6-33. The master application can avoid this procedure if the control subsystem application in flash takes care of this by itself or if the user's application code, downloaded to M0 RAM. zero-initializes all the control subsystem RAMs before using them.

6.7.2 Master Subsystem Application Procedure to Initialize the Control Subsystem RAM Using IPC

As explained in Section 6.6.2, all the control subsystem RAM, except for M0 RAM must be zero-Initialized before they are used by the control subsystem applications. This is to avoid un-wanted RAM ECC errors because of uninitialized RAM locations.

MOTC IPC commands should be used by the master subsystem application to set bits in control subsystem registers to zero-Initialize all control subsystem memories. The following code shows the procedure in detail.

Below are some defines that the code uses for ease of implementation. These registers are explicitly defined and described in the *Internal Memory* chapter. The functions below use the IPCLITE driver library software to send IPC commands to C-Boot ROM. The IPCLITE driver library is released as part of header files.

Disclaimer:-
Copyright (c) 2008-2011 Texas Instruments Incorporated. All rights reserved.
Software License Agreement Texas Instruments (TI) is supplying this software for use solely and exclusively on TI's microcontroller products. The software is owned by TI and/or its suppliers, and is protected under applicable copyright laws. You may not combine this software with "viral" open-source software in order to form a larger program.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITH ALL FAULTS. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. TI SHALL NOT, UNDER ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

```
#define CCORE_M0M1_CTOM_MSG_RAM_INIT_REG_ADDR           0x4920
#define CCORE_M0M1_CTOM_MSG_RAM_INIT_DONE_REG_ADDR      0x4930
#define CCORE_L0L4_RAM_INIT_REG_ADDR                    0x4922
#define CCORE_L0_L4_RAM_INIT_DONE_REG_ADDR              0x4932

#define CCORE_L0_RAM_INIT_DONE_BIT                      0x01
#define CCORE_L1_RAM_INIT_DONE_BIT                      0x04
#define CCORE_L2_RAM_INIT_DONE_BIT                      0x10
#define CCORE_L3_RAM_INIT_DONE_BIT                      0x40
#define CCORE_L0_RAM_INIT_BIT                           0x01
#define CCORE_L1_RAM_INIT_BIT                           0x04
#define CCORE_L2_RAM_INIT_BIT                           0x10
#define CCORE_L3_RAM_INIT_BIT                           0x40
#define CCORE_M1_RAM_INIT_BIT                           0x04
#define CCORE_CTOM_MSG_RAM_INIT_BIT                     0x10
#define CCORE_M1_RAM_INIT_DONE_BIT                      0x04
#define CCORE_CTOM_MSG_RAM_INIT_DONE_BIT                0x10

/*Function Name:  master_ram_init_control_m1_msgram_memories
* Description:-  Function called to Zero-Initialize control subsystem M1 and Message RAM memories
*         using MTOCIPC commands to C-Boot ROM
*
*/
void master_ram_init_control_m1_msgram_memories()
{
     unsigned int ii = 0;

      //RAM INIT for M1, and CTOM MsgRAM - M0 RAM INIT is done by C-BootROM
        IPCLiteMtoCSetBits_Protected( IPC_FLAG1,  CCORE_M0M1_CTOM_MSG_RAM_INIT_REG_ADDR ,
```


Copyright © 2012–2019, Texas Instruments Incorporated

```
        (CCORE_M1_RAM_INIT_BIT | CCORE_CTOM_MSG_RAM_INIT_BIT), IPC_LENGTH_32_BITS, IPC_FLAG32);


      //wait until C-BootROM acks
        while(HWREG(MTOCIPC_BASE + IPC_O_MTOCIPCFLG) and IPC_FLAG1);

      //CHECK IF pass or fail
        if(HWREG(MTOCIPC_BASE + IPC_O_MTOCIPCFLG) and IPC_FLAG32)
       {//still set - so command failed.
                            while(1)
                            {
                                    for(ii = 0; ii < 0xDEADC0DE; ii++);
                                    //ERROR – HAVE an ERROR INDICATOR here
                            }

       }
        //READ RAM_INIT_DONE register check do
        {
               IPCLiteMtoCDataRead(IPC_FLAG1, CCORE_M0M1_CTOM_MSG_RAM_INIT_DONE_REG_ADDR,
IPC_LENGTH_32_BITS, IPC_FLAG32);

                //wait until C-BootROM acks
                while(HWREG(MTOCIPC_BASE + IPC_O_MTOCIPCFLG) and IPC_FLAG1);

                //CHECK IF pass or fail
                if(HWREG(MTOCIPC_BASE + IPC_O_MTOCIPCFLG) and IPC_FLAG32)
                {//still set - so command failed.
                    while(1)
                                {
                                    for(ii = 0; ii < 0xDEADC0DE; ii++);
                                 //ERROR – HAVE an ERROR INDICATOR here
                                }

                    }
                    else
                    {
                                if(HWREG(MTOCIPC_BASE + IPC_O_MTOCIPCDATAR) and
(CCORE_CTOM_MSG_RAM_INIT_DONE_BIT|CCORE_M1_RAM_INIT_DONE_BIT))
                                {
                                        //RAM_INIT completed - do nothing :-)
                                        break;
                                 }
                                else
                                {
                                         //RAM_INIT not done yet, so wait for more time and read
ram init done register.
                                          //GIVE some cycles delay until CCORE performs RAM INIT
                                          for(ii =0 ; ii < 2048; ii++);
                                          continue;
                                }
                          }
        }while(1);   //do-loop-end

} //function-end


/*Function Name:  master_ram_init_control_L0_L3_memories
* Description:-  Function called to Zero-Initialize control subsystem L0, L1, L2, L3 RAM memories
*        using MTOCIPC commands to C-Boot ROM
*
*/

void master_ram_init_control_L0_L3_memories()
{
               unsigned int ii = 0;

               //RAM INIT for L0,L1,L2,L3
```


Copyright © 2012–2019, Texas Instruments Incorporated

```
                IPCLiteMtoCSetBits_Protected(IPC_FLAG1, CCORE_L0L4_RAM_INIT_REG_ADDR,
(CCORE_L0_RAM_INIT_BIT| CCORE_L1_RAM_INIT_BIT| CCORE_L2_RAM_INIT_BIT| CCORE_L3_RAM_INIT_BIT),
IPC_LENGTH_32_BITS, IPC_FLAG32);

                //wait until C-BootROM acks
                while(HWREG(MTOCIPC_BASE + IPC_O_MTOCIPCFLG) and IPC_FLAG1);

                //CHECK IF pass or fail
                if(HWREG(MTOCIPC_BASE + IPC_O_MTOCIPCFLG) and IPC_FLAG32)
                {//still set - so command failed.
                                //error - blink SLOOOOOOOOWWW
                                    while(1)
                                    {
                                       for(ii = 0; ii < 0xDEADC0DE; ii++);

                      //ERROR – HAVE an ERROR INDICATOR here
                                    }

                }

                //WAIT for RAM_INIT done of above memories

                do
                {
                                IPCLiteMtoCDataRead(IPC_FLAG1, CCORE_L0_L4_RAM_INIT_DONE_REG_ADDR,
IPC_LENGTH_32_BITS, IPC_FLAG32);

                                //wait until C-BootROM acks
                                while(HWREG(MTOCIPC_BASE + IPC_O_MTOCIPCFLG) and IPC_FLAG1);

                                //CHECK IF pass or fail
                                if(HWREG(MTOCIPC_BASE + IPC_O_MTOCIPCFLG) and IPC_FLAG32)
                                {//still set - so command failed.
                                                while(1)
                                                    {
                                                       for(ii = 0; ii < 0xDEADC0DE; ii++);

                                     //ERROR – HAVE an ERROR INDICATOR here
                                                    }

                                }
                                else
                                {
                                                if(HWREG(MTOCIPC_BASE + IPC_O_MTOCIPCDATAR) and
(CCORE_L0_RAM_INIT_DONE_BIT| CCORE_L1_RAM_INIT_DONE_BIT| CCORE_L2_RAM_INIT_DONE_BIT|
CCORE_L3_RAM_INIT_DONE_BIT ))
                                                {
                                                                //RAM_INIT completed - do nothing :-)
                                                                break;
                                                }
                                                else
                                                {
                                                                //RAM_INIT not done yet, so wait for
more time and read ram init done register.
                                                                //GIVE some cycles delay until CCORE
performs RAM INIT
                                                                for(ii =0 ; ii < 2048; ii++);
                                                                continue;
                                                }
                                }
                }while(1);


}
```


Copyright © 2012–2019, Texas Instruments Incorporated

6.8 Application Notes to Use Boot ROM

This section provides guidelines on how to boot-load master and control subsystems on Concerto devices.

6.8.1 How to Send Boot Data to M-Boot ROM

This section explains how to build a bootable image using Code Composer Studio™ and how to transfer the image to the master subsystem RAM, using the M-Boot ROM peripheral bootloaders.

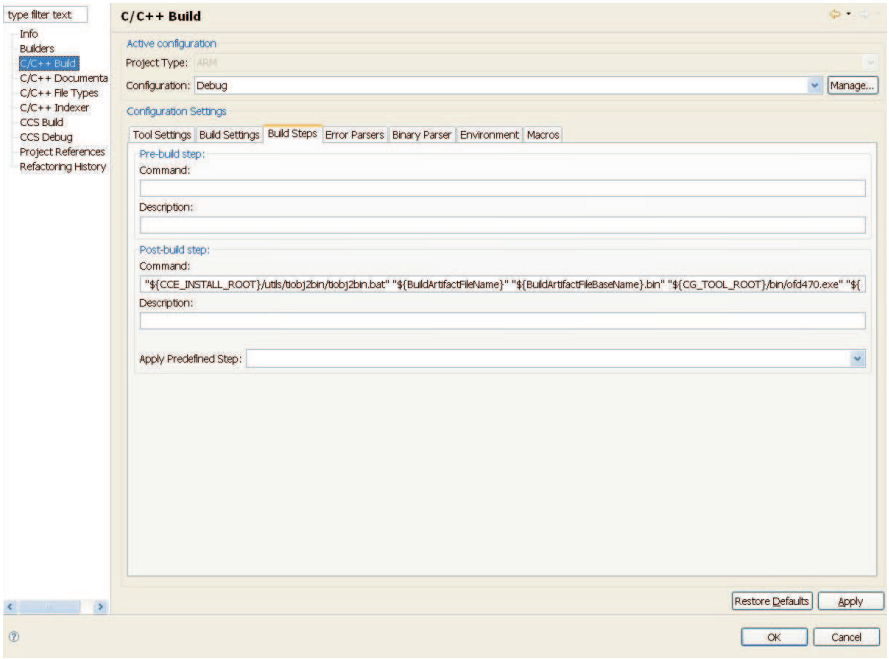
6.8.1.1 Building the Binary Image for an Application – Using Code Composr Studio

When using the CCS IDE to build code to RUN on the Concerto platform, the binary bootable image is automatically generated by the following command in the Post-Build step of project settings.

post-build step:"${CCE_INSTALL_ROOT}/utils/tiobj2bin/tiobj2bin.bat" "${BuildArtifactFileName}" "${BuildArtifactFileBaseName}.bin" "${CG_TOOL_ROOT}/bin/ofd470.exe" "${CG_TOOL_ROOT}/bin/hex470.exe" "${CCE_INSTALL_ROOT}/utils/tiobj2bin/mkhex4bin.exe"

Following is the snapshot of ‘project build properties dialogue’ in CCS which has the above post-build step.

Figure 6-34. Build a Binary Image for Bootload Using M-BOOT ROM



The following utilities that are installed during the default CCS install are used to generate the final binary image file that can be transferred to M-Boot ROM.

C:/Program Files/Texas Instruments/ccsv4/utils/tiobj2bin/tiobj2bin.bat

C:/Program Files/Texas Instruments/ccsv4/tools/compiler/tms470/bin/ofd470.exe

C:/Program Files/Texas Instruments/ccsv4/tools/compiler/tms470/bin/hex470.exe

C:/Program Files/Texas Instruments/ccsv4/utils/tiobj2bin/mkhex4bin.exe

In the above Post-Build command, the "${BuildArtifactFileName}" is the project COFF file that the user is currently building. This COFF file is given as input to the above scripts to generate a final plain binary image that can be used (as shown in the following section), to send data to M-Boot ROM.

Copyright © 2012–2019, Texas Instruments Incorporated

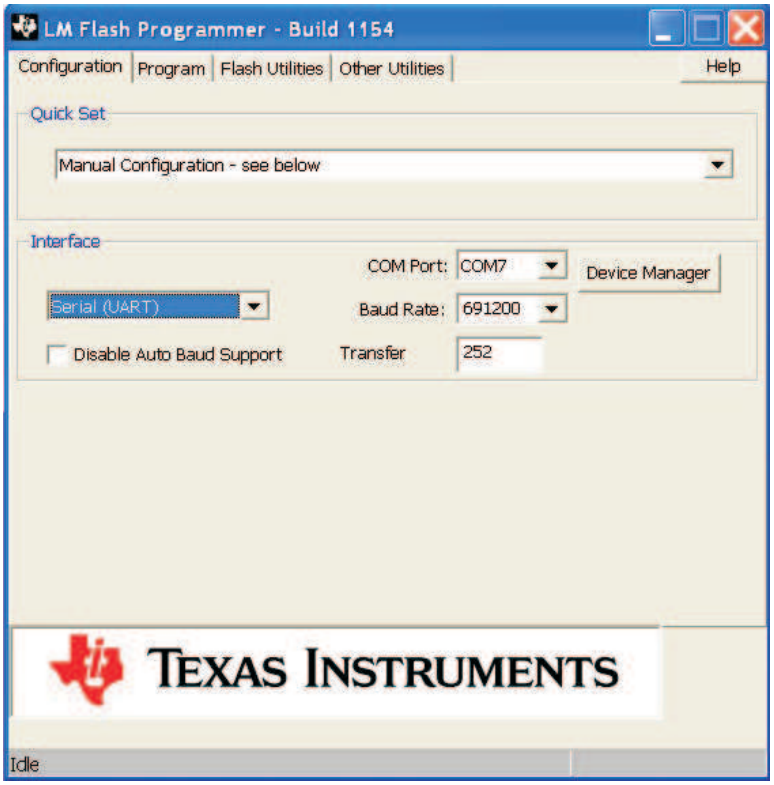
6.8.1.2 Using the LM FLASH Programmer to Transfer In Image to Concerto Devices from a PC

As explained earlier, M-Boot ROM Serial, EMAC and CAN boot modes are compatible to respective boot modes in Stellaris devices.

The snapshots below show how the LM Flash programmer utility available on the web can be used to send data to M-Boot ROM via UART0 and EMAC peripherals from a PC. In these snapshots, LM flash programmer is running on the same host which is transferring data to the device. Note that on this device, only bootloading of data to RAM is supported. Even though the LM flash programmer utility is used on PC/HOST, data is only loaded to the master subsystem RAM(s).

1. Start LM flash programmer and select Manual configuration as shown below:

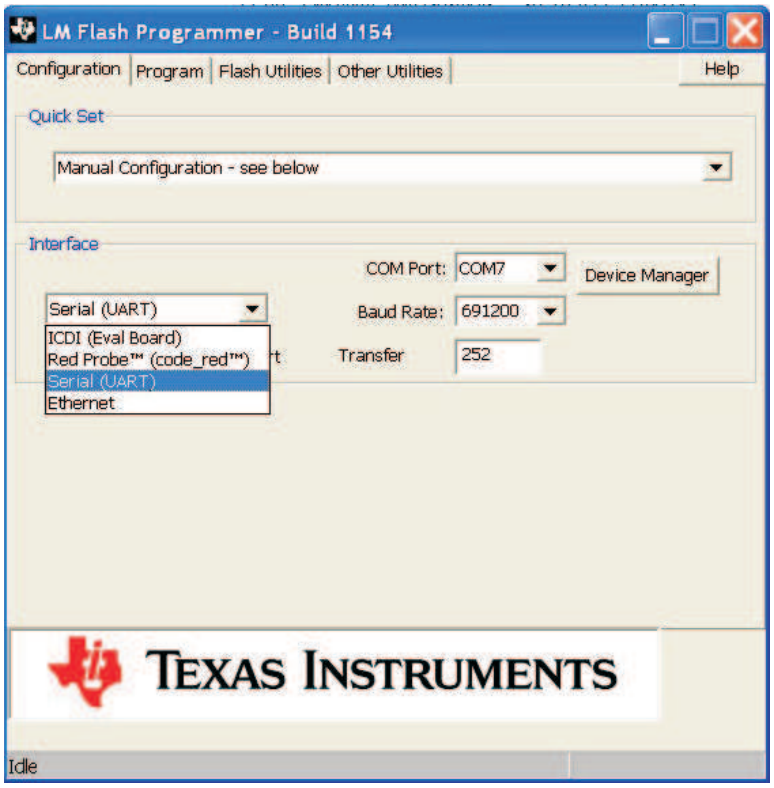
Figure 6-35. LM FLASH Programmer Configuration Screen



2. Select either UART or Ethernet interface as shown below, depending on which interface he is using to boot load data.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-36. LM FLASH Programmer Interface Selection Screen

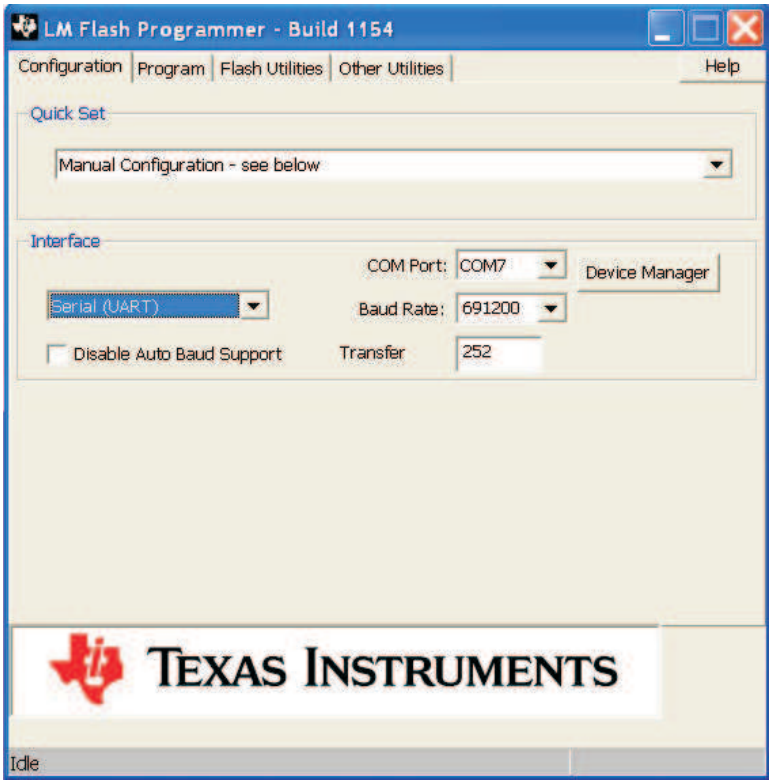


6.8.1.2.1 *Using the LM FLASH Programmer to Send Data to the M-Boot ROM UART0 Interface – Serial Bootload Option*

1. Select the UART interface and configure the HOST COM Port Baud Rate as needed.

Copyright © 2012–2019, Texas Instruments Incorporated

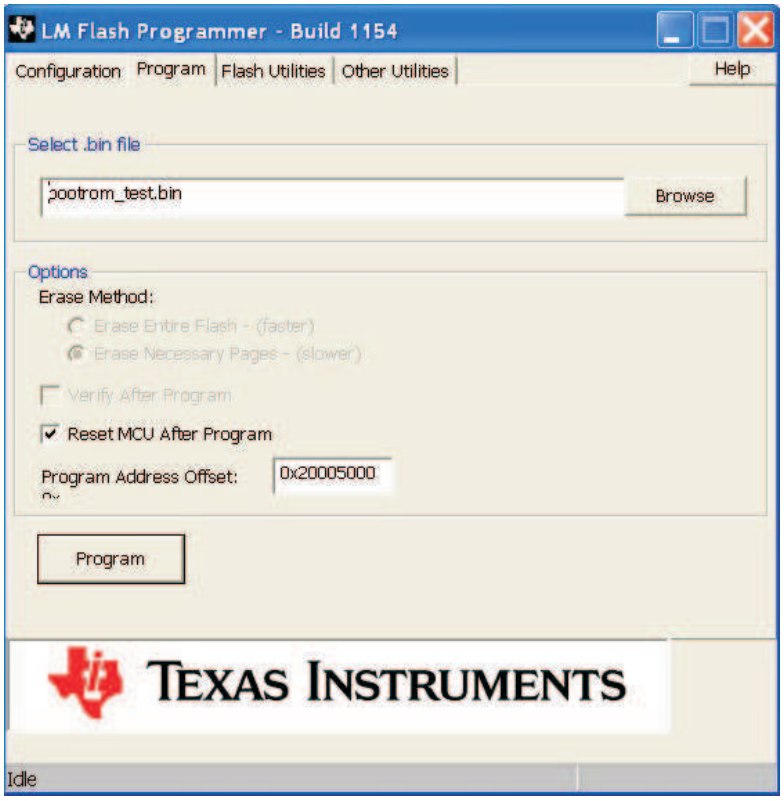
Figure 6-37. LM FLASH Programmer Serial Interface Configuration Screen



2. Select the Program tab as shown below and select the bin file built as explained in Section 6.8.1.1. The Program Address Offset in the snapshot below is the LOAD address for M-Boot ROM to load the application. The same address is the Application START address. The 'Reset MCU After Program' option has to be selected for the application to be automatically started after load is complete. This option doesn't reset the MCU, but will send the RESET packet to the M-Boot ROM bootloader which will start executing the application code just downloaded.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-38. LM FLASH Programmer Binary Image Selection Screen

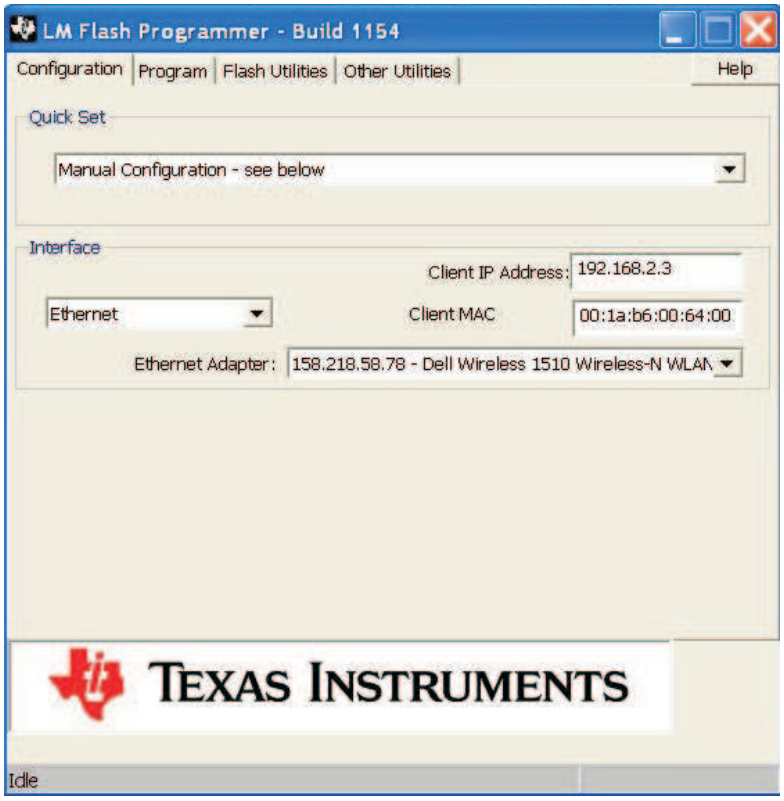


6.8.1.2.2 *Using the LM FLASH Programmer to Send Data to the M-BOOT ROM EMAC Interface – Ethernet Bootload Option*

1. Select the Ethernet option as shown below. The Client MAC field should be filled in with the EMAC IF of the Concerto device, and the Client IP Address field should be filled with a proper IP address that the user wants to assign to the device.

Copyright © 2012–2019, Texas Instruments Incorporated

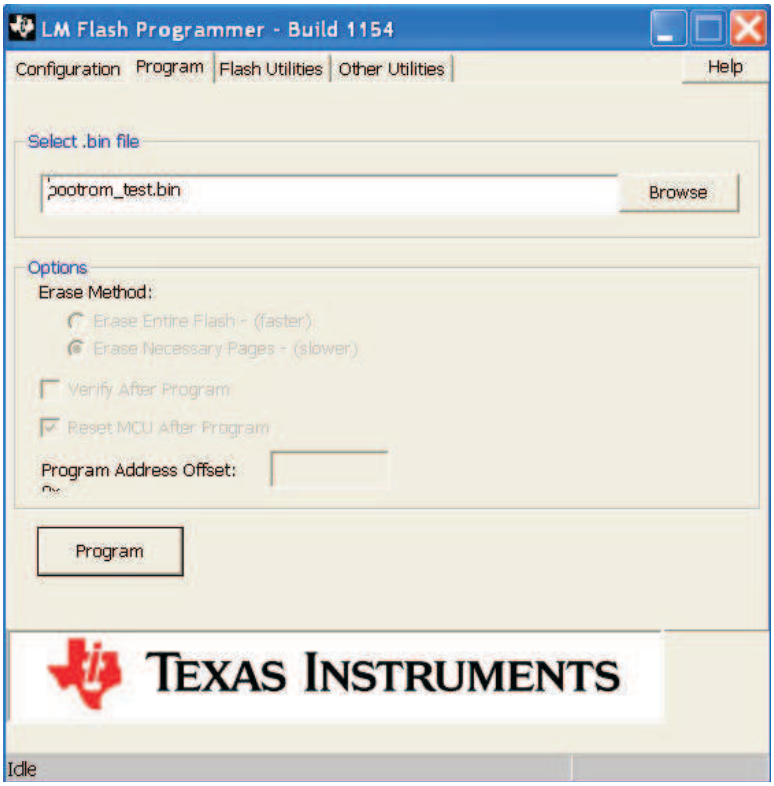
Figure 6-39. LM FLASH Programmer EMAC Interface Selection Screen



2. Select the program tab as shown below and select the binary image file built as shown in Section 6.8.1.1.
   Note: There is no way for the user to control where in the master subsystem RAM, the application should get loaded. It is fixed at M_BOOT_ROM_RAM_ENTRY_POINT as explained in Section 6.5.7.5

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 6-40. FLASH Programmer EMAC Bootload Binary Image Selection Screen



6.8.2 C-Boot ROM: Building the Boot Table

This chapter explains how to generate the data stream and boot table required for the bootloader.

6.8.2.1 The C2000 Hex Utility

To use the features of the bootloader, you must generate a data stream and boot table as described in Section 6.6.14.1. The hex conversion utility tool, included with the 28x code generation tools, can generate the required data stream including the required boot table. This section describes the hex2000 utility. An example of a file conversion performed by hex2000 is described in Section 6.8.2.2.

The hex utility supports creation of the boot table required for the SCI, SPI, I2C, eCAN, and parallel I/O loaders. That is, the hex utility adds the required information to the file such as the key value, reserved bits, entry point, address, block start address, block length and terminating value. The contents of the boot table vary slightly depending on the boot mode and the options selected when running the hex conversion utility. The actual file format required by the host (ASCII, binary, hex, etc.) will differ from one specific application to another and some additional conversion may be required.

To build the boot table, follow these steps:

1. **Assemble or compile the code.**

   This creates the object files that will then be used by the linker to create a single output file.

2. **Link the file.**

   The linker combines all of the object files into a single output file in common object file format (COFF). The specified linker command file is used by the linker to allocate the code sections to different memory blocks. Each block of the boot table data corresponds to an initialized section in the COFF file. Uninitialized sections are not converted by the hex conversion utility. The following options may be useful:

   The linker -m option can be used to generate a map file. This map file will show all of the sections that were created, their location in memory and their length. It can be useful to check this file to make sure

Copyright © 2012–2019, Texas Instruments Incorporated

that the initialized sections are where you expect them to be.

The linker -w option is also very useful. This option will tell you if the linker has assigned a section to a memory region on its own. For example, if you have a section in your code called ramfuncs.

3. **Run the hex conversion utility.**

   Choose the appropriate options for the desired boot mode and run the hex conversion utility to convert the COFF file produced by the linker to a boot table.

See the *TMS320C28x Assembly Language Tools User's Guide* (SPRU513) and the *TMS320C28x Optimizing C/C++ Compiler User's Guide* (SPRU514) for more information on the compiling and linking process.

Table 6-31 summarizes the hex conversion utility options available for the bootloader. See the *TMS320C28x Assembly Language Tools User's Guide* (SPRU513) for a detailed description of the hex2000 operations used to generate a boot table. Updates will be made to support the I2C boot. See the Codegen release notes for the latest information.

Table 6-31. Bootloader Options

Option	Description
-boot	Convert all sections into bootable form (use instead of a SECTIONS directive)
-sci8	Specify the source of the bootloader table as the SCI-A port, 8-bit mode
-spi8	Specify the source of the bootloader table as the SPI-A port, 8-bit mode
-gpio8	Specify the source of the bootloader table as the GPIO port, 8-bit mode
-gpio16	Specify the source of the bootloader table as the GPIO port, 16-bit mode
-bootorg value	Specify the source address of the bootloader table
-lospcp value	Specify the initial value for the LOSPCP register. This value is used only for the spi8 boot table format and ignored for all other formats. If the value is greater than 0x7F, the value is truncated to 0x7F.
-spibrr value	Specify the initial value for the SPIBRR register. This value is used only for the spi8 boot table format and ignored for all other formats. If the value is greater than 0x7F, the value is truncated to 0x7F.
-e value	Specify the entry point at which to begin execution after boot loading. The value can be an address or a global symbol. This value is optional. The entry point can be defined at compile time using the linker -e option to assign the entry point to a global symbol. The entry point for a C program is normally _c_int00 unless defined otherwise by the -e linker option.
-i2c8	Specify the source of the bootloader table as the I2C-A port, 8-bit
-i2cpsc value	Specify the value for the I2CPSC register. This value will be loaded and take effect after all I2C options are loaded, prior to reading data from the EEPROM. This value will be truncated to the least significant eight bits and should be set to maintain an I2C module clock of 7-12 MHz.
-i2cclkh value	Specify the value for the I2CCLKH register. This value will be loaded and take effect after all I2C options are loaded, prior to reading data from the EEPROM.
-i2cclkl value	Specify the value for the I2CCLKL register. This value will be loaded and take effect after all I2C options are loaded, prior to reading data from the EEPROM.

6.8.2.2 Example: Preparing a COFF File For eCAN Bootloading

This section shows how to convert a COFF file into a format suitable for CAN based bootloading. This example assumes that the host sending the data stream is capable of reading an ASCII hex format file. An example COFF file named GPIO34TOG.out has been used for the conversion.

Build the project and link using the -m linker option to generate a map file. Examine the .map file produced by the linker. The information shown in Example 6-5 has been copied from the example map file (GPIO34TOG.map). This shows the section allocation map for the code. The map file includes the following information:

- **Output Section**

  This is the name of the output section specified with the SECTIONS directive in the linker command file.

- **Origin**

  The first origin listed for each output section is the starting address of that entire output section. The following origin values are the starting address of that portion of the output section.

Copyright © 2012–2019, Texas Instruments Incorporated

- **Length**

  The first length listed for each output section is the length for that entire output section. The following length values are the lengths associated with that portion of the output section.

- **Attributes/input sections**

  This lists the input files that are part of the section or any value associated with an output section.

See the *TMS320C28x Assembly Language Tools User's Guide* (SPRU513) for detailed information on generating a linker command file and a memory map.

All sections shown in Example 6-5 that are initialized need to be loaded into the DSP in order for the code to execute properly. In this case, the codestart, ramfuncs, .cinit, myreset and .text sections need to be loaded. The other sections are uninitialized and will not be included in the loading process. The map file also indicates the size of each section and the starting address. For example, the .text section has 0x155 words and starts at 0x3FA000.

Example 6-5. GPIO34TOG Map File

```
output                                              attributes/
section         page     origin      length       input sections
--------        ----   ----------  ----------  ----------------
codestart
                  0     00000000     00000002
                        00000000     00000002   DSP280x_CodeStartBranch.obj (codestart)
.pinit            0     00000002     00000000

.switch           0     00000002     00000000   UNINITIALIZED

ramfuncs          0     00000002     00000016
                        00000002     00000016   DSP280x_SysCtrl.obj (ramfuncs)

.cinit            0     00000018     00000019
                        00000018     0000000e   rts2800_ml.lib : exit.obj (.cinit)
                        00000026     0000000a                  : _lock.obj (.cinit)
                        00000030     00000001   --HOLE-- [fill = 0]

myreset           0     00000032     00000002
                        00000032     00000002   DSP280x_CodeStartBranch.obj (myreset)

IQmath            0     003fa000     00000000   UNINITIALIZED

.text             0     003fa000     00000155
                        003fa000     00000046   rts2800_ml.lib : boot.obj (.text)
```

To load the code using the CAN bootloader, the host must send the data in the format that the bootloader understands. That is, the data must be sent as blocks of data with a size, starting address followed by the data. A block size of 0 indicates the end of the data. The HEX2000.exe utility can be used to convert the COFF file into a format that includes this boot information. The following command syntax has been used to convert the application into an ASCII hex format file that includes all of the required information for the bootloader:

Example 6-6. HEX2000.exe Command Syntax

```
C: HEX2000 GPIO34TOG.OUT -boot -gpio8 -a


Where:
- boot   Convert all sections into bootable form.
- gpio8  Use the GPIO in 8-bit mode data format. The eCAN
         uses the same data format as the GPIO in 8-bit mode.
- a      Select ASCII-Hex as the output format.
```


Copyright © 2012–2019, Texas Instruments Incorporated

The command line shown in Example 6-6 will generate an ASCII-Hex output file called GPIO34TOG.a00, whose contents are explained in Example 6-7. This example assumes that the host will be able to read an ASCII hex format file. The format may differ for your application. . Each section of data loaded can be tied back to the map file described in Example 6-5. After the data stream is loaded, the boot ROM will jump to the Entrypoint address that was read as part of the data stream. In this case, execution will begin at 0x3FA0000.

Copyright © 2012–2019, Texas Instruments Incorporated

Example 6-7. GPIO34TOG Data Stream

```
AA 08                                       ;Keyvalue
00 00 00 00 00 00 00 00                     ;8 reserved words
00 00 00 00 00 00 00 00
3F 00 00 A0                                 ;Entrypoint 0x003FA000
02 00                                       ;Load 2 words - codestart section
00 00 00 00                                 ;Load block starting at 0x000000
7F 00 9A A0                                 ;Data block 0x007F, 0xA09A
16 00                                       ;Load 0x0016 words - ramfuncs section
00 00 02 00                                 ;Load block starting at 0x000002
22 76 1F 76 2A 00 00 1A 01 00 06 CC F0      ;Data = 0x7522, 0x761F etc...
FF 05 50 06 96 06 CC FF F0 A9 1A 00 05
06 96 04 1A FF 00 05 1A FF 00 1A 76 07
F6 00 77 06 00
55 01                                       ;Load 0x0155 words - .text section
3F 00 00 A0                                 ;Load block starting at 0x003FA000
AD 28 00 04 69 FF 1F 56 16 56 1A 56 40      ;Data = 0x28AD, 0x4000 etc...
29 1F 76 00 00 02 29 1B 76 22 76 A9 28
18 00 A8 28 00 00 01 09 1D 61 C0 76 18
00 04 29 0F 6F 00 9B A9 24 01 DF 04 6C
04 29 A8 24 01 DF A6 1E A1 F7 86 24 A7
06 .. ..
.. .. ..
.. .. ..
FC 63 E6 6F
19 00 ;Load 0x0019 words - .cinit section
00 00 18 00                                 ;Load block starting at 0x000018
FF FF 00 B0 3F 00 00 00 FE FF 02 B0 3F      ;Data = 0xFFFF, 0xB000 etc...
00 00 00 00 00 FE FF 04 B0 3F 00 00 00
00 00 FE FF .. .. ..
.. .. ..
3F 00 00 00
02 00                                       ;Load 0x0002 words - myreset section
00 00 32 00                                 ;Load block starting at 0x000032
00 00 00 00                                 ;Data = 0x0000, 0x0000
00 00                                       ;Block size of 0 - end of data
```


Copyright © 2012–2019, Texas Instruments Incorporated

C28 Enhanced Pulse Width Modulator (ePWM) Module

The enhanced pulse width modulator (ePWM) peripheral is a key element in controlling many of the power electronic systems found in both commercial and industrial equipments. These systems include digital motor control, switch mode power supply control, uninterruptible power supplies (UPS), and other forms of power conversion. The ePWM peripheral performs a digital to analog (DAC) function, where the duty cycle is equivalent to a DAC analog value; it is sometimes referred to as a Power DAC.

This chapter guide is applicable for ePWM Type 2. See the *TMS320x28xx, 28xxx DSP Peripheral Reference Guide* (SPRU566) for a list of all devices with an ePWM module of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

Topic **Page**

7.1 Introduction **631**
7.2 ePWM Submodules **641**
7.3 Applications to Power Topologies **707**
7.4 Registers **733**

Copyright © 2012–2019, Texas Instruments Incorporated

7.1 Introduction

This chapter includes an overview of the module and information about each of its submodules:

- Time-Base Module
- Counter Compare Module
- Action Qualifier Module
- Dead-Band Generator Module
- PWM Chopper (PC) Module
- Trip Zone Module
- Event Trigger Module
- Digital Compare Module

ePWM Type 2 is fully compatible to the Type 1 module. Type 2 has the following enhancements in addition to the Type 1 features:

- **New Registers added at upper page offset**

  Additional registers are required for new features on ePWM Type 2. ePWM register set for each module has been expanded from 64 16-bit words to 128 16-bit words.
- **High Resolution Dead-Band Capability**

  High-Resolution capability is added to Dead-band RED and FED in half-cycle clocking mode.
- **Dead-Band Generator Module Enhancements**

  ePWM Type 2 has features to enable both RED and FED on either PWM outputs. Provides increased deadband with 14-bit counters and Dead-band / Dead-band high-resolution registers are shadowed
- **High Resolution Extension available on ePWMxB outputs**

  Provides the ability to enable high-resolution period and duty cycle control on ePWMxB outputs. This is discussed in more detail in the *HRPWM Reference Guide*.
- **Counter Compare Module Enhancements**

  ePWM Type 2 allows Interrupts and SOC events to be generated by additional counter compares CMPC and CMPD.
- **Event Trigger Module Enhancements**

  Prescaling logic to issue interrupt requests and ADC start of conversion expanded upto every 15 events.Allows Software initialization of event counters on SYNC event.
- **Digital Compare Module Enhancements**

  Digital Compare Trip Select logic [DCTRIPSEL] has upto 12 external trip sources selected by GPTRIP logic in addition to an ability to OR all of them (up to 14 [external & internal sources]) to create respective DCxEVTs .
- **Simultaneous Writes to TBPRD and CMPx Registers**

  This feature allows writes to TBPRD, CMPA:CMPAHR, CMPB:CMPBHR, CMPC and CMPD of any ePWM module to be tied to any other ePWM module, and also allows all ePWM modules to be tied to a particular ePWM module if desired.
- **Shadow to Active Load on SYNC of TBPRD and CMP Registers**

  This feature supports simultaneous writes of TBPRD and CMPA/B/C/D registers.

An effective PWM peripheral must be able to generate complex pulse width waveforms with minimal CPU overhead or intervention. It needs to be highly programmable and very flexible while being easy to understand and use. The ePWM unit described here addresses these requirements by allocating all needed timing and control resources on a per PWM channel basis. Cross coupling or sharing of resources has been avoided; instead, the ePWM is built up from smaller single channel modules with separate resources that can operate together as required to form a system. This modular approach results in an orthogonal architecture and provides a more transparent view of the peripheral structure, helping users to understand its operation quickly.

Copyright © 2012–2019, Texas Instruments Incorporated

In this document the letter x within a signal or module name is used to indicate a generic ePWM instance on a device. For example, output signals EPWMxA and EPWMxB refer to the output signals from the ePWMx instance. Thus, EPWM1A and EPWM1B belong to ePWM1 and likewise EPWM4A and EPWM4B belong to ePWM4.

7.1.1 Submodule Overview

The ePWM module represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB. Multiple ePWM modules are instanced within a device as shown in Figure 7-1. Each ePWM instance is identical with one exception. Some instances include a hardware extension that allows more precise control of the PWM outputs. This extension is the high-resolution pulse width modulator (HRPWM) and is described in the device-specific *High-Resolution Pulse Width Modulator (HRPWM) Reference Guide*. See the device-specific data manual to determine which ePWM instances include this feature. Each ePWM module is indicated by a numerical value starting with 1. For example ePWM1 is the first instance and ePWM3 is the 3rd instance in the system and ePWMx indicates any instance.

The ePWM modules are chained together via a clock synchronization scheme that allows them to operate as a single system when required. Additionally, this synchronization scheme can be extended to the capture peripheral modules (eCAP). The number of modules is device-dependent and based on target application needs. Modules can also operate stand-alone.

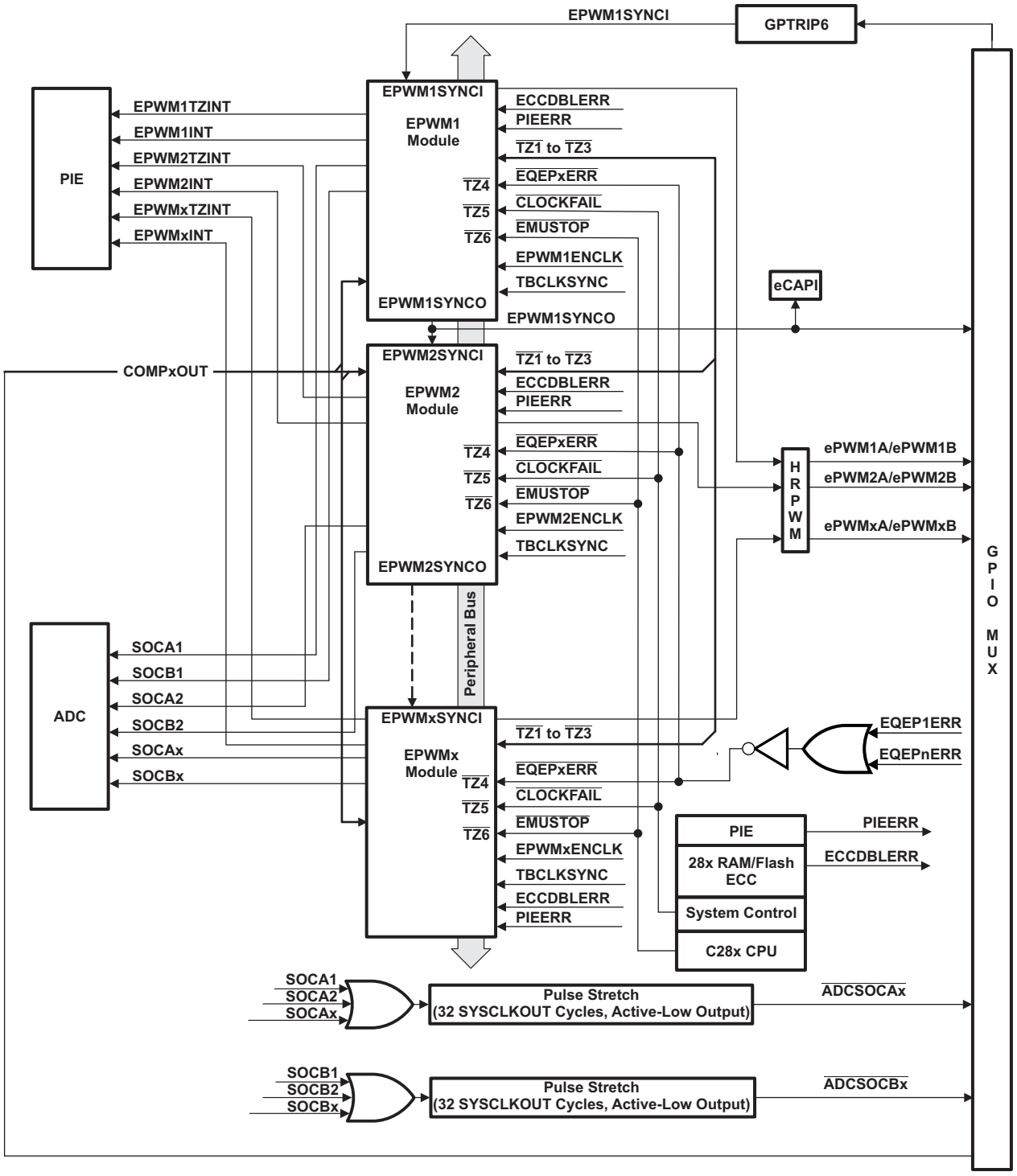
Each ePWM module supports the following features:

- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations:
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software.
- Programmable phase-control support for lag or lead operation relative to other ePWM modules.
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis.
- Dead-band generation with independent rising and falling edge delay control.
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions.
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs.
- All events can trigger both CPU interrupts and ADC start of conversion (SOC)
- Programmable event prescaling minimizes CPU overhead on interrupts.
- PWM chopping by high-frequency carrier signal, useful for pulse transformer gate drives.

Each ePWM module is connected to the input/output signals shown in Figure 7-1. The signals are described in detail in subsequent sections.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-1. Multiple ePWM Modules



A This signal exists only on devices with an eQEP module.

The order in which the ePWM modules are connected may differ from what is shown in Figure 7-1. See Section 7.2.2.3.3 for the synchronization scheme for a particular device. Each ePWM module consists of eight submodules and is connected within a system via the signals shown in Figure 7-2.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-2. Submodules and Signal Connections for an ePWM Module

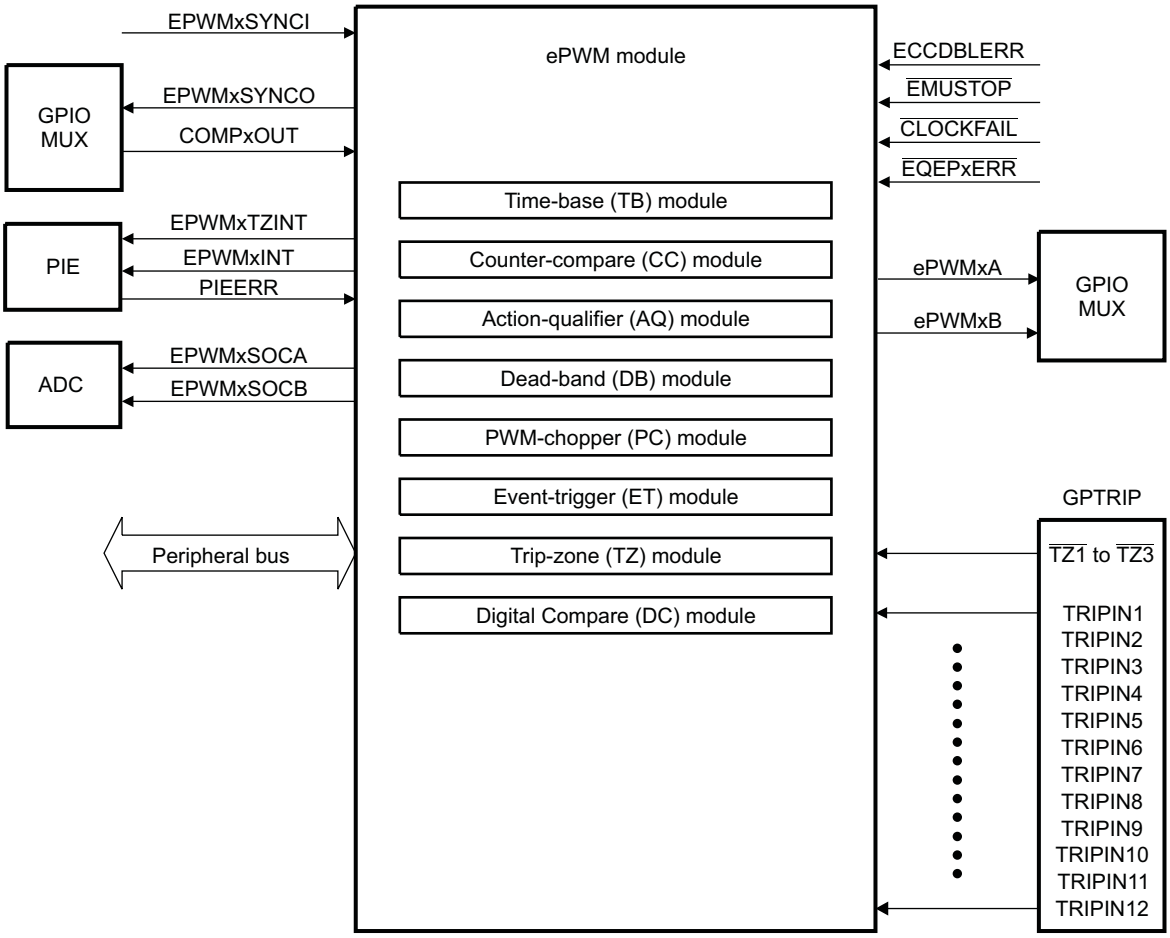


Figure 7-3 shows more internal details of a single ePWM module. The main signals used by the ePWM module are:

- **PWM output signals (EPWMxA and EPWMxB).**

  The PWM output signals are made available external to the device through the GPIO peripheral described in the system control and interrupts guide for your device.

- **Trip-zone signals ($\overline{TZ1}$ to $\overline{TZ6}$).**

  These input signals alert the ePWM module of fault conditions external to the ePWM module. Each module on a device can be configured to either use or ignore any of the trip-zone signals. The $\overline{TZ1}$ to $\overline{TZ3}$ trip-zone signals can be configured as asynchronous inputs through the GPIO peripheral using the GPTRIP logic, Refer to Figure 7-48. $\overline{TZ4}$ is connected to an inverted EQEPx error signal (EQEPxERR) which can be generated from any one of the EQEP module (for those devices with an EQEP module). $\overline{TZ5}$ is connected to the system clock fail logic, and $\overline{TZ6}$ is connected to the $\overline{EMUSTOP}$ output from the CPU. This allows you to configure a trip action when the clock fails or the CPU halts.

- **Time-base synchronization input (EPWMxSYNCI) and output (EPWMxSYNCO) signals**.

  The synchronization signals daisy chain the ePWM modules together. Each module can be configured via GPTRIP6 to either use or ignore its synchronization input. The clock synchronization input and output signal are brought out to pins only for ePWM1 (ePWM module #1). The synchronization output for ePWM1 (EPWM1SYNCO) is also connected to the SYNCI of the first enhanced capture module (eCAP1). Refer to Time-Base Counter Synchronization Scheme 4 in this chapter.

- **ADC start-of-conversion signals (EPWMxSOCA and EPWMxSOCB).**

  Each ePWM module has two ADC start of conversion signals . Any ePWM module can trigger a start of conversion. Whichever event triggers the start of conversion is configured in the Event-Trigger submodule of the ePWM.

Copyright © 2012–2019, Texas Instruments Incorporated

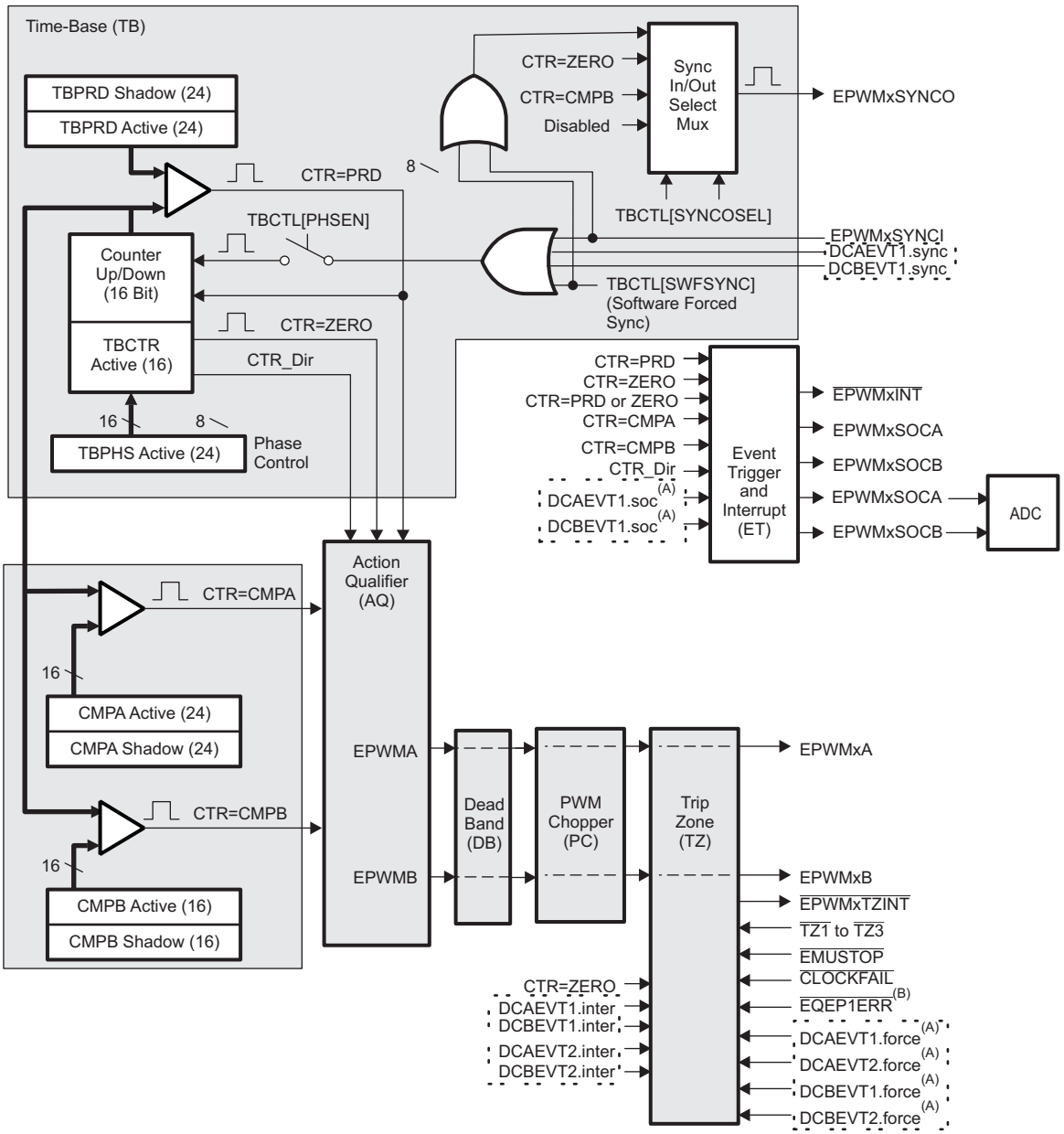
- **Comparator output signals (COMPxOUT).**

  Output signals from the comparator module can be fed through the GPIO peripheral [GPTRIP logic] to one or all of the 12 trip inputs [TRIPIN1 - TRIPIN12] and in conjunction with the trip zone signals can generate digital compare events.

- **Peripheral Bus**

  The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the ePWM register file.

Figure 7-3. ePWM Submodules and Critical Internal Signal Interconnects



A These events are generated by the type 2 ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs [For Example: COMPxOUT and $\overline{TZ}$ signals].

B This signal exists only on devices with eQEP module

Figure 7-3 also shows the key internal submodule interconnect signals. Each submodule is described in detail in its respective section.

Copyright © 2012–2019, Texas Instruments Incorporated

7.1.2 Register Mapping

The complete ePWM module control and status register set is grouped by submodule as shown in Table 7-1. Each register set is duplicated for each instance of the ePWM module. The start address for each ePWM register file instance on a device is specified in the appropriate data manual.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-1. ePWM Module Control and Status Register Set Grouped by Submodule

Name	Offset [1]	Size (x16)	Shadow	EALLOW	Description
					Time-Base Submodule Registers
TBCTL	0x00	1	No		Time-Base Control Register
TBSTS	0x01	1	No		Time-Base Status Register
TBPHSHR	0x02	1	No		Extension for HRPWM Phase Register [2]
TBPHS	0x03	1	No		Time-Base Phase Register
TBCTR	0x04	1	No		Time-Base Counter Register
TBPRD	0x05	1	Yes		Time-Base Period Register
TBPRDHR	0x06	1	Yes		Time Base Period High Resolution Register [3]
					Counter-Compare Submodule Registers
CMPCTL	0x07	1	No		Counter-Compare Control Register
CMPAHR	0x08	1	Yes		Extension for HRPWM Counter-Compare A Register [2]
CMPA	0x09	1	Yes		Counter-Compare A Register
CMPB	0x0A	1	Yes		Counter-Compare B Register
					Action-Qualifier Submodule Registers
AQCTLA	0x0B	1	No		Action-Qualifier Control Register for Output A (EPWMxA)
AQCTLB	0x0C	1	No		Action-Qualifier Control Register for Output B (EPWMxB)
AQSFRC	0x0D	1	No		Action-Qualifier Software Force Register
AQCSFRC	0x0E	1	Yes		Action-Qualifier Continuous S/W Force Register Set
					Dead-Band Generator Submodule Registers
DBCTL	0x0F	1	No		Dead-Band Generator Control Register
DBRED	0x10	1	No		Dead-Band Generator Rising Edge Delay Count Register
DBFED	0x11	1	No		Dead-Band Generator Falling Edge Delay Count Register
					Trip-Zone Submodule Registers
TZSEL	0x12	1		Yes	Trip-Zone Select Register
TZDCSEL	0x13	1		Yes	Trip Zone Digital Compare Select Register
TZCTL	0x14	1		Yes	Trip-Zone Control Register [3]
TZEINT	0x15	1		Yes	Trip-Zone Enable Interrupt Register [3]
TZFLG	0x16	1			Trip-Zone Flag Register [3]
TZCLR	0x17	1		Yes	Trip-Zone Clear Register [3]
TZFRC	0x18	1		Yes	Trip-Zone Force Register [3]
					Event-Trigger Submodule Registers
ETSEL	0x19	1			Event-Trigger Selection Register
ETPS	0x1A	1			Event-Trigger Pre-Scale Register
ETFLG	0x1B	1			Event-Trigger Flag Register
ETCLR	0x1C	1			Event-Trigger Clear Register
ETFRC	0x1D	1			Event-Trigger Force Register
					PWM-Chopper Submodule Registers
PCCTL	0x1E	1			PWM-Chopper Control Register
					High-Resolution Pulse Width Modulator (HRPWM) Extension Registers
HRCNFG	0x20	1		Yes	HRPWM Configuration Register [2] [3]
HRPWR	0x21	1		Yes	HRPWM Power Register [3] [4]
HRMSTEP	0x26	1		Yes	HRPWM MEP Step Register [3] [4]

(1) Locations not shown are reserved.
(2) These registers are only available on ePWM instances that include the high-resolution PWM extension. Otherwise these locations are reserved. These registers are described in the *High-Resolution Pulse Width Modulator (HRPWM)* section of this manual. See the device specific data manual to determine which instances include the HRPWM.
(3) EALLOW protected registers as described in the specific device version of the *System Control and Interrupts Reference Guide*.
(4) These registers only exist in the ePWM1 register space. They cannot be accessed from any other ePWM module's register space.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-1. ePWM Module Control and Status Register Set Grouped by Submodule (continued)

Name	Offset [1]	Size (x16)	Shadow	EALLOW	Description
HRCNFG2	0x27	1		Yes	HRPWM Configuration 2 Register [2] [3]
HRPCTL	0x28	1		Yes	High Resolution Period Control Register[3]
TBPRDHRM	0x2A	1	Writes		Time Base Period High Resolution Register Mirror[3]
TBPRDM	0x2B	1	Writes		Time Base Period Register Mirror
CMPAHRM	0x2C	1	Writes		Compare A High-Resolution Register Mirror[3]
CMPAM	0x2D	1	Writes		Compare A Register Mirror
					Digital Compare Event Registers
DCTRIPSEL	0x30	1		Yes	Digital Compare Trip Select Register
DCACTL	0x31	1		Yes	Digital Compare A Control Register
DCBCTL	0x32	1		Yes	Digital Compare B Control Register
DCFCTL	0x33	1		Yes	Digital Compare Filter Control Register
DCCAPCTL	0x34	1		Yes	Digital Compare Capture Control Register
DCFOFFSET	0x35	1	Writes		Digital Compare Filter Offset Register
DCFOFFSETCNT	0x36	1			Digital Compare Filter Offset Counter Register
DCFWINDOW	0x37	1			Digital Compare Filter Window Register
DCFWINDOWCNT	0x38	1			Digital Compare Filter Window Counter Register
DCCAP	0x39	1	Yes		Digital Compare Counter Capture Register

The type 2 Enhanced Pulse Width Modulator (ePWM) features demand additional bits on the register set. To achieve this on this device, the ePWM register set for each module has been expanded from 64 16-bit words to 128 16-bit words. This document refers to these additional register sets as upper 64 16-bit words or, simply, upper page. On the upper page offset listed in Table 7-2:

- 0x40-0x5F contains new configuration registers along with CMPBHR:CMPB for shadow read/shadow write.
- 0x60-0x6F contains frequently-used high-resolution registers along with new CMPC and CMPD registers.
- 0x70-0x7F contains dual-mapped frequently-used control registers from lower page in order to accommodate minimal DP pointer switching.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-2. ePWM Module Control and Status Register Set Grouped by Submodule (Upper Page)

Name	Offset[1]	Size (x16)	Shadow	EALLOW	Description
TBCTL2	0x40	1			Time Base Control Register 2
CMPCTL2	0x41	1			Counter Compare Control Register 2
AQCTLR	0x47	1			Action Qualifier Control Register
CMPBHR	0x4A	1	Yes		Compare B High-Resolution Register
DCAHTRIPSEL	0x4C	1		Yes	Digital Compare AH Trip Select
DCALTRIPSEL	0x4D	1		Yes	Digital Compare AL Trip Select
DCBHTRIPSEL	0x4E	1		Yes	Digital Compare BH Trip Select
DCBLTRIPSEL	0x4F	1		Yes	Digital Compare BL Trip Select
ETINTPS	0x50	1			Event-Trigger Interrupt Pre-Scale Register
ETSOCPS	0x51	1			Event-Trigger SOC Pre-Scale Register
ETCNTINITCTL	0x52	1			Event-Trigger Counter Initialization Control Register
ETCNTINIT	0x53	1			Event-Trigger Counter Initialization Register
EPWMXLINK	0x5E - 0x5F	2			EPWMx Link Register
TBPHSHRM	0x60	1			Time Base Phase High-Resolution Mirror Register[2]
TBPHSM	0x61	1			Time Base Phase Mirror Register[2]
TBPRDHRM2	0x62	1	Yes		Time Base Period High-Resolution Mirror 2 Register[3]
TBPRDM2	0x63	1	Yes		Time Base Period Mirror 2 Register[3]
CMPAHRM2	0x64	1	Yes		Counter-Compare A High-Resolution Mirror 2 Register[3]
CMPAM2	0x65	1	Yes		Counter-Compare A Mirror 2 Register[3]
CMPBHRM	0x66	1	Yes		Counter-Compare B Mirror High-Resolution Register[2]
CMPBM	0x67	1	Yes		Counter-Compare B Mirror Register[2]
CMPC	0x69	1	Yes		Counter-Compare C Register
CMPD	0x6B	1	Yes		Counter-Compare D Register
DBREDHR	0x6C	1	Yes		Dead-Band Generator Rising Edge Delay High-Resolution Mirror Register
DBREDM	0x6D	1	Yes		Dead-Band Generator Rising Edge Delay Count Mirror Register[2]
DBFEDHR	0x6E	1	Yes		Dead-Band Generator Falling Edge Delay High Resolution Register
DBFEDM	0x6F	1	Yes		Dead-Band Generator Falling Edge Delay Count Register[2]
ETCLRM	0x70	1			Event Trigger Clear Register[2]
TZCLRM	0x71	1		Yes	Trip Zone Clear Register[2]
AQCTLAM	0x73	1	Yes		Action Qualifier Control Register For Output A[2]
AQCTLBM	0x74	1	Yes		Action Qualifier Control Register For Output B[2]
AQSFRCM	0x75	1			Action Qualifier Software Force Register[2]
AQCSFRCM	0x76	1	Yes		Action Qualifier Continuous S/W Force Register[2]

(1) Locations not shown are reserved.
(2) These registers are dual-mapped at lower page.
(3) These registers are triple-mapped at lower page.

The CMPA, CMPB, CMPAHR, CMPBHR, TBPRD, DBRED, TBPRDHR, DBREDHR, DBFED, DBFEDHR registers are mirrored in the register map (mirror registers include a "-M" or "-M2" suffix). Note in the tables below that in immediate mode and Shadow mode reads from these mirror registers result in the active value of the register or a TI internal test value. The below tables are arranged such that first set of columns indicate the primary register and their behaviour in shadow/immediate mode with respect to a Read or Write operation. The second set of columns in the same row indicate the multiple mirror register [if exisiting] and their behaviour in shadow/immediate mode with respect to a Read or Write operation.

In Immediate Mode:

Copyright © 2012–2019, Texas Instruments Incorporated

Register	Offset	Write	Read	Register	Offset	Write	Read
TBPRDHR	0x06	Active	Active	TBPRDHRM / TBPRDHRM2	0x2A / 0x62	Active	TI_Internal / Active
TBPRD	0x05	Active	Active	TBPRDM /TBPRDM2	0x2B / 0x63	Active	Active / Active
CMPAHR	0x08	Active	Active	CMPAHRM / CMPAHRM2	0x2C / 0x64	Active	TI_Internal / Active
CMPA	0x09	Active	Active	CMPAM / CMPAM2	0x2D / 0x65	Active	Active / Active
CMPBHR	0x4A	Active	Active	CMPBHRM	0x66	Active	TI_Internal
CMPB	0x0A	Active	Active	CMPBM	0x67	Active	Active
DBREDHR	0x50	Active	TI_Internal				
DBRED	0x10	Active	Active	DBREDM	0x51	Active	Active
DBREDHR	0x52	Active	TI_Internal				
DBFED	0x11	Active	Active	DBFEDM	0x53	Active	Active
CMPC	0x69	Active	Active				
CMPD	0x6B	Active	Active				

In Shadow Mode:

Register	Offset	Write	Read	Register	Offset	Write	Read
TBPRDHR	0x06	Shadow	Shadow	TBPRDHRM / TBPRDHRM2	0x2A / 0x62	Shadow	Active / Shadow
TBPRD	0x05	Shadow	Shadow	TBPRDM /TBPRDM2	0x2B / 0x63	Shadow	Active / Shadow
CMPAHR	0x08	Shadow	Shadow	CMPAHRM / CMPAHRM2	0x2C / 0x64	Shadow	Active / Shadow
CMPA	0x09	Shadow	Shadow	CMPAM / CMPAM2	0x2D / 0x65	Shadow	Active / Shadow
CMPBHR	0x4A	Shadow	Shadow	CMPBHRM	0x66	Shadow	Active
CMPB	0x0A	Shadow	Shadow	CMPBM	0x67	Shadow	Active
DBREDHR	0x50	Shadow	Shadow				
DBRED	0x10	Shadow	Shadow	DBREDM	0x51	Shadow	Shadow
DBREDHR	0x52	Shadow	Shadow				
DBFED	0x11	Shadow	Shadow	DBFEDM	0x53	Shadow	Shadow
CMPC	0x69	Shadow	Shadow				
CMPD	0x6B	Shadow	Shadow				

Copyright © 2012–2019, Texas Instruments Incorporated

7.2 ePWM Submodules

Eight submodules are included in every ePWM peripheral. Each of these submodules performs specific tasks that can be configured by software.

7.2.1 Overview

Table 7-3 lists the eight key submodules together with a list of their main configuration parameters. For example, if you need to adjust or control the duty cycle of a PWM waveform, then you should see the counter-compare submodule in Section 7.2.3 for relevant details.

Table 7-3. Submodule Configuration Parameters

Submodule	Configuration Parameter or Option
Time-base (TB)	• Scale the time-base clock (TBCLK) relative to the system clock (SYSCLKOUT). • Configure the PWM time-base counter (TBCTR) frequency or period. • Set the mode for the time-base counter: – count-up mode: used for asymmetric PWM – count-down mode: used for asymmetric PWM – count-up-and-down mode: used for symmetric PWM • Configure the time-base phase relative to another ePWM module. • Synchronize the time-base counter between modules through hardware or software. • Configure the direction (up or down) of the time-base counter after a synchronization event. • Simultaneous writes to the TBPRD registers on all PWM's corresponding to the configuration on EPWMXLINK. • Configure how the time-base counter will behave when the device is halted by an emulator. • Specify the source for the synchronization output of the ePWM module: – Synchronization input signal – Time-base counter equal to zero – Time-base counter equal to counter-compare B (CMPB) – No output synchronization signal generated.
Counter-compare (CC)	• Specify the PWM duty cycle for output EPWMxA and/or output EPWMxB • Specify the time at which switching events occur on the EPWMxA or EPWMxB output • Specify the programmable delay for interrupt and SOC generation with additional comparators • Simultaneous writes to the CMPA, CMPB, CMPC, CMPD registers on all PWM's corresponding to the configuration on EPWMXLINK.
Action-qualifier (AQ)	• Specify the type of action taken when a time-base or counter-compare submodule event occurs: – No action taken – Output EPWMxA and/or EPWMxB switched high – Output EPWMxA and/or EPWMxB switched low – Output EPWMxA and/or EPWMxB toggled • Force the PWM output state through software control • Configure and control the PWM dead-band through software
Dead-band (DB)	• Control of traditional complementary dead-band relationship between upper and lower switches • Specify the output rising-edge-delay value • Specify the output falling-edge delay value • Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification. • Option to enable half-cycle clocking for double resolution. • Allow ePWMxB phase shifting with respect to the ePWMxA output.
PWM-chopper (PC)	• Create a chopping (carrier) frequency. • Pulse width of the first pulse in the chopped pulse train. • Duty cycle of the second and subsequent pulses. • Bypass the PWM-chopper module entirely. In this case the PWM waveform is passed through without modification.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-3. Submodule Configuration Parameters (continued)

Submodule	Configuration Parameter or Option
Trip-zone (TZ)	• Configure the ePWM module to react to one, all, or none of the trip-zone signals or digital compare events. • Specify the tripping action taken when a fault occurs: – Force EPWMxA and/or EPWMxB high – Force EPWMxA and/or EPWMxB low – Force EPWMxA and/or EPWMxB to a high-impedance state – Configure EPWMxA and/or EPWMxB to ignore any trip condition. • Configure how often the ePWM will react to each trip-zone signal: – One-shot – Cycle-by-cycle • Enable the trip-zone to initiate an interrupt. • Bypass the trip-zone module entirely. • Programmable option for Cycle-by-cycle trip clear
Event-trigger (ET)	• Enable the ePWM events that will trigger an interrupt. • Enable ePWM events that will trigger an ADC start-of-conversion event. • Specify the rate at which events cause triggers (every occurrence or every 2nd or up to 15th occurrence) • Poll, set, or clear event flags
Digital-compare (DC)	• Enables comparator (COMP) module outputs and trip zone signals which are configured using the GPTRIP logic to create events and filtered events • Specify event-filtering options to capture TBCTR counter or generate blanking window

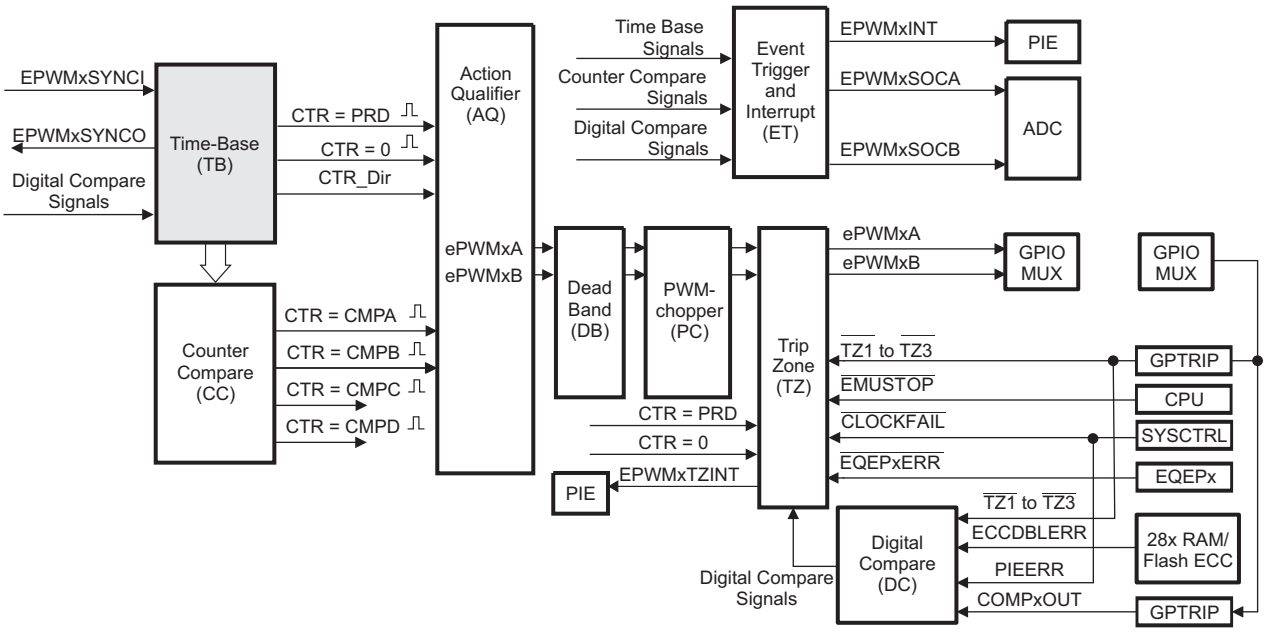
Code examples are provided in the remainder of this document that show how to implement various ePWM module configurations. These examples use the constant definitions in the device *EPwm_defines.h* file in the device-specific header file and peripheral examples software package.

Copyright © 2012–2019, Texas Instruments Incorporated

7.2.2 Time-Base (TB) Submodule

Each ePWM module has its own time-base submodule that determines all of the event timing for the ePWM module. Built-in synchronization logic allows the time-base of multiple ePWM modules to work together as a single system. Figure 7-4 illustrates the time-base module's place within the ePWM.

Figure 7-4. Time-Base Submodule



7.2.2.1 Purpose of the Time-Base Submodule

You can configure the time-base submodule for the following:

- Specify the ePWM time-base counter (TBCTR) frequency or period to control how often events occur.
- Manage time-base synchronization with other ePWM modules.
- Maintain a phase relationship with other ePWM modules.
- Set the time-base counter to count-up, count-down, or count-up-and-down mode.
- Generate the following events:
  - CTR = PRD: Time-base counter equal to the specified period (TBCTR = TBPRD) .
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00).
- Configure the rate of the time-base clock; a prescaled version of the CPU system clock (SYSCLKOUT). This allows the time-base counter to increment/decrement at a slower rate.

Copyright © 2012–2019, Texas Instruments Incorporated

7.2.2.2 Controlling and Monitoring the Time-Base Submodule

Table 7-4 shows the registers used to control and monitor the time-base submodule.

Table 7-4. Time-Base Submodule Registers

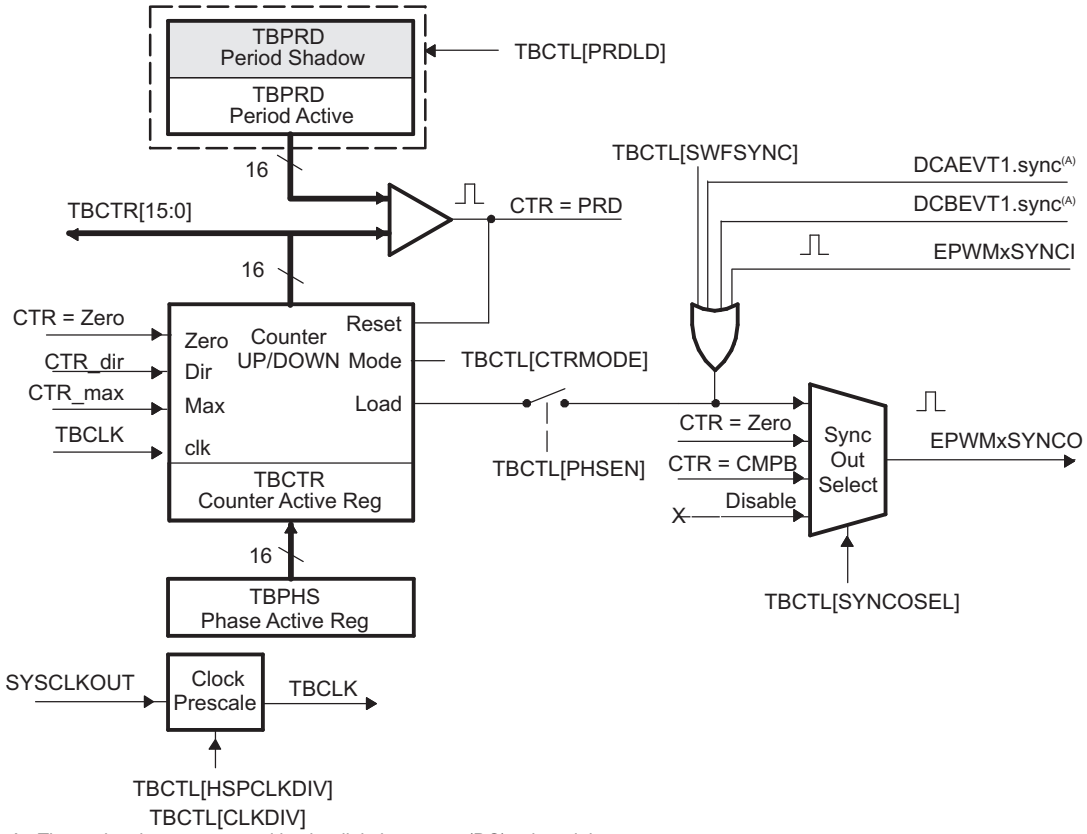
Register Name	Address Offset	Shadowed	Description
TBCTL	0x00	No	Time-Base Control Register
TBSTS	0x01	No	Time-Base Status Register
TBPHSHR	0x02	No	HRPWM Extension Phase Register(1)
TBPHS	0x03	No	Time-Base Phase Register
TBCTR	0x04	No	Time-Base Counter Register
TBPRD	0x05	Yes	Time-Base Period Register
TBPRDHR	0x06	Yes	HRPWM Extension Period Register(1)
TBPRDHRM	0x2A	Yes	HRPWM Time-Base Period Extension Mirror Register(1)
TBPRDM	0x2B	Yes	HRPWM Extension Period Mirror Register(1)
TBCTL2	0x40	No	Time Base Control Register 2
TBPHSHRM	0x60	No	Time Base Phase High Resolution Mirror Register(1)
TBPHSM	0x61	No	Time Base Phase Mirror Register
TBPRDHRM2	0x62	Yes	Time Base Period High Resolution Mirror 2 Register(1)
TBPRDM2	0x63	Yes	Time Base Period Mirror 2 Register
EPWMXLINK	0x5E - 0x5F	No	EPWMx Link Register

(1) This register is available only on ePWM instances that include the high-resolution extension (HRPWM). On ePWM modules that do not include the HRPWM this location is reserved. This register is described in the device-specific *High-Resolution Pulse Width Modulator (HRPWM)* section of this manual. Refer to the device-specific data manual to determine which ePWM instances include this feature.

Copyright © 2012–2019, Texas Instruments Incorporated

The block diagram in Figure 7-5 shows the critical signals and registers of the time-base submodule. Table 7-5 provides descriptions of the key signals associated with the time-base submodule.

Figure 7-5. Time-Base Submodule Signals and Registers



A. These signals are generated by the digital compare (DC) submodule.

A These events are generated by the type 2 ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs [For Example: COMPxOUT and $\overline{TZ}$ signals].

Table 7-5. Key Time-Base Signals

Signal	Description
EPWMxSYNCI	Time-base synchronization input. Input pulse used to synchronize the time-base counter with the counter of ePWM module earlier in the synchronization chain. An ePWM peripheral can be configured to use or ignore this signal. For the first ePWM module (EPWM1) this signal comes from a device pin via GPTRIP6. For subsequent ePWM modules this signal is passed from another ePWM peripheral. For example, EPWM2SYNCI is generated by the ePWM1 peripheral, EPWM3SYNCI is generated by ePWM2 and so forth. See Section 7.2.2.3.3 for information on the synchronization order of a particular device.
EPWMxSYNCO	Time-base synchronization output. This output pulse is used to synchronize the counter of an ePWM module later in the synchronization chain. The ePWM module generates this signal from one of three event sources: 1. EPWMxSYNCI (Synchronization input pulse) 2. CTR = Zero: The time-base counter equal to zero (TBCTR = 0x00). 3. CTR = CMPB: The time-base counter equal to the counter-compare B (TBCTR = CMPB) register.
CTR = PRD	Time-base counter equal to the specified period. This signal is generated whenever the counter value is equal to the active period register value. That is when TBCTR = TBPRD.
CTR = Zero	Time-base counter equal to zero This signal is generated whenever the counter value is zero. That is when TBCTR equals 0x00.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-5. Key Time-Base Signals (continued)

Signal	Description
CTR = CMPB	Time-base counter equal to active counter-compare B register (TBCTR = CMPB).
	This event is generated by the counter-compare submodule and used by the synchronization out logic
CTR_dir	Time-base counter direction.
	Indicates the current direction of the ePWM's time-base counter. This signal is high when the counter is increasing and low when it is decreasing.
CTR_max	Time-base counter equal max value. (TBCTR = 0xFFFF)
	Generated event when the TBCTR value reaches its maximum value. This signal is only used only as a status bit
TBCLK	Time-base clock.
	This is a prescaled version of the system clock (SYSCLKOUT) and is used by all submodules within the ePWM. This clock determines the rate at which time-base counter increments or decrements.

7.2.2.3 Calculating PWM Period and Frequency

The frequency of PWM events is controlled by the time-base period (TBPRD) register and the mode of the time-base counter. Figure 7-6 shows the period (T_{pwm}) and frequency (F_{pwm}) relationships for the up-count, down-count, and up-down-count time-base counter modes when when the period is set to 4 (TBPRD = 4). The time increment for each step is defined by the time-base clock (TBCLK) which is a prescaled version of the system clock (SYSCLKOUT).

The time-base counter has three modes of operation selected by the time-base control register (TBCTL):

- **Up-Down-Count Mode:**

  In up-down-count mode, the time-base counter starts from zero and increments until the period (TBPRD) value is reached. When the period value is reached, the time-base counter then decrements until it reaches zero. At this point the counter repeats the pattern and begins to increment.

- **Up-Count Mode:**

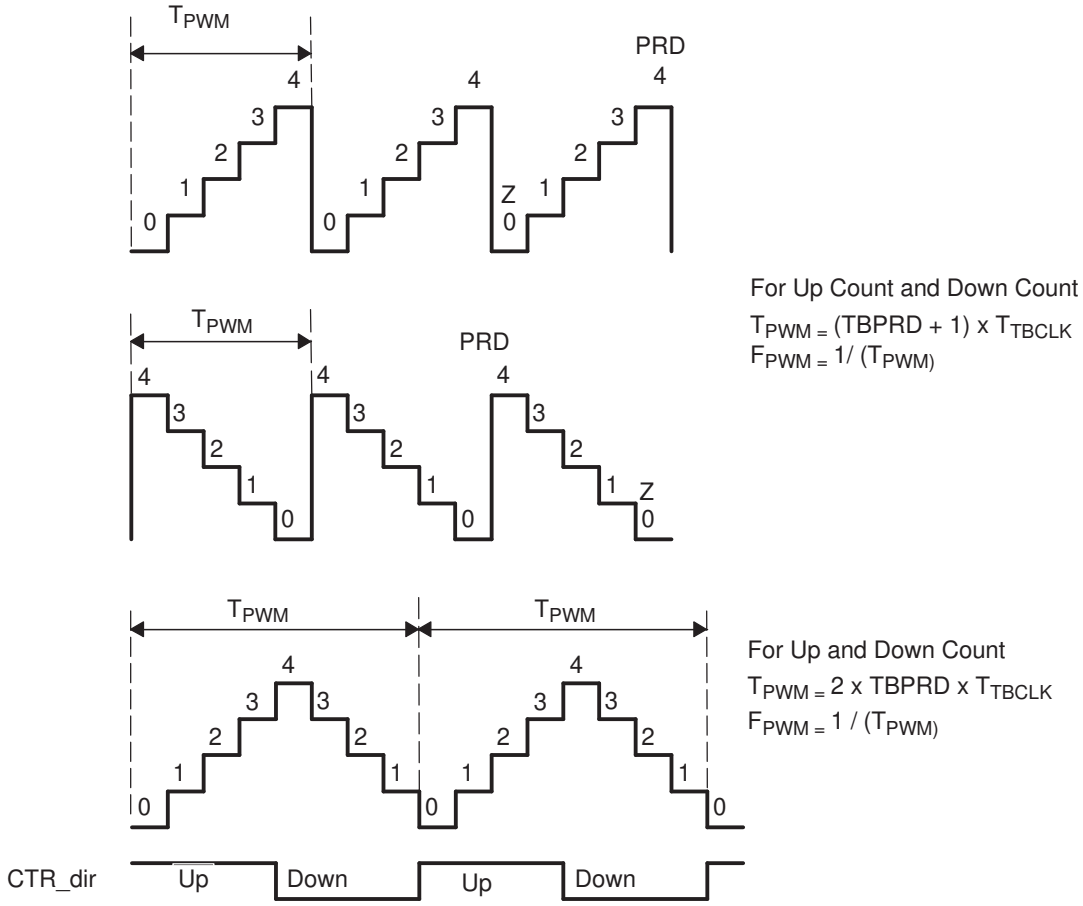
  In this mode, the time-base counter starts from zero and increments until it reaches the value in the period register (TBPRD). When the period value is reached, the time-base counter resets to zero and begins to increment once again.

- **Down-Count Mode:**

  In down-count mode, the time-base counter starts from the period (TBPRD) value and decrements until it reaches zero. When it reaches zero, the time-base counter is reset to the period value and it begins to decrement once again.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-6. Time-Base Frequency and Period



7.2.2.3.1 Time-Base Period Shadow Register

The time-base period register (TBPRD) has a shadow register. Shadowing allows the register update to be synchronized with the hardware. The following definitions are used to describe all shadow registers in the ePWM module:

- **Active Register**

  The active register controls the hardware and is responsible for actions that the hardware causes or invokes.

- **Shadow Register**

  The shadow register buffers or provides a temporary holding location for the active register. It has no direct effect on any control hardware. At a strategic point in time the shadow register's content is transferred to the active register. This prevents corruption or spurious operation due to the register being asynchronously modified by software.

The memory address of the shadow period register is the same as the active register. Which register is written to or read from is determined by the TBCTL[PRDLD] bit. This bit enables and disables the TBPRD shadow register as follows:

- **Time-Base Period Shadow Mode:**

  The TBPRD shadow register is enabled when TBCTL[PRDLD] = 0. Reads from and writes to the TBPRD memory address go to the shadow register. The shadow register contents are transferred to the active register (TBPRD (Active) ← TBPRD (shadow)) when the time-base counter equals zero (TBCTR = 0x00) and/or a sync event as determined by the TBCTL2[PRDLDSYNC] bit. The PRDLDSYNC bit is valid only if TBCTL[PRDLD] = 0. By default the TBPRD shadow register is enabled. The sources for the SYNC input is explained in Time-Base Counter Synchronization section

Copyright © 2012–2019, Texas Instruments Incorporated

of this chapter

- **Time-Base Period Immediate Load Mode:**

  If immediate load mode is selected (TBCTL[PRDLD] = 1), then a read from or a write to the TBPRD memory address goes directly to the active register.

7.2.2.3.2 Time-Base Clock Synchronization

The TBCLKSYNC bit in the peripheral clock enable registers allows all users to globally synchronize all enabled ePWM modules to the time-base clock (TBCLK). When set, all enabled ePWM module clocks are started with the first rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescalers for each ePWM module must be set identically.

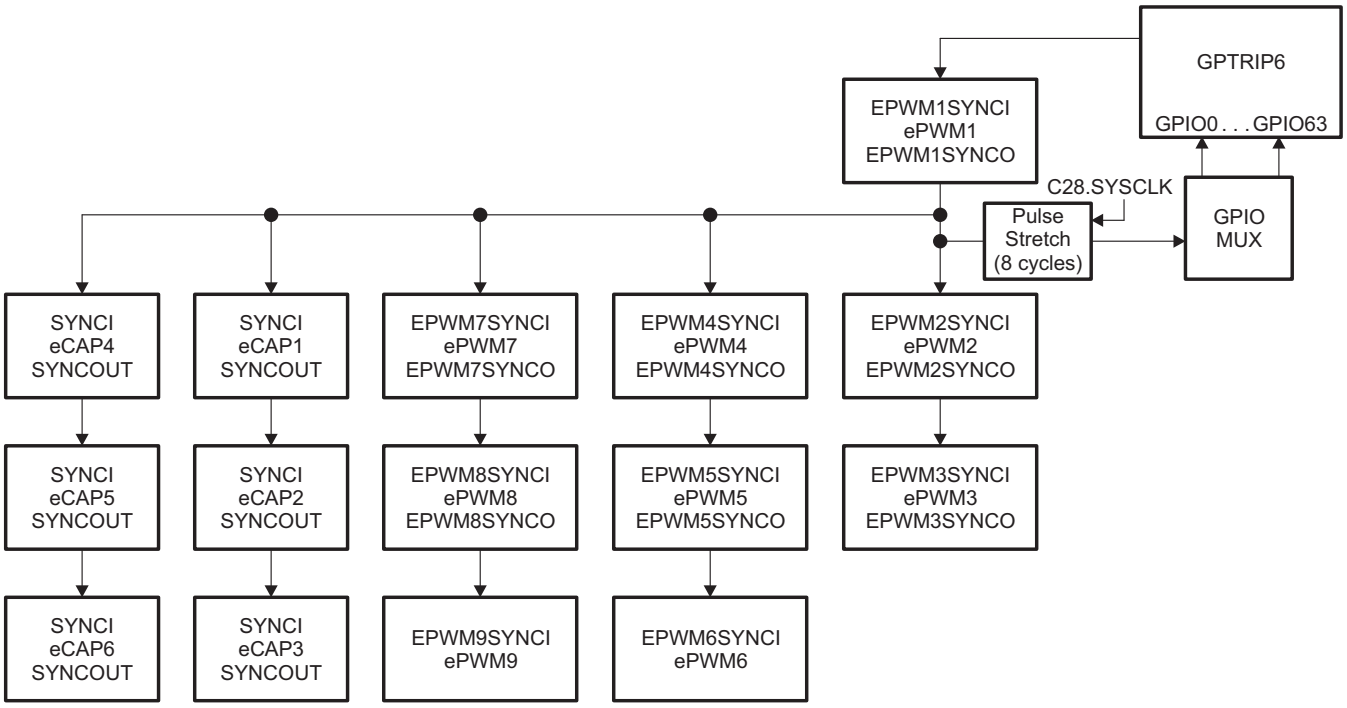
The proper procedure for enabling ePWM clocks is as follows:

1. Enable ePWM module clocks in the PCLKCRx register
2. Set TBCLKSYNC= 0
3. Configure ePWM modules
4. Set TBCLKSYNC=1

7.2.2.3.3 Time-Base Counter Synchronization

A time-base synchronization scheme connects all of the ePWM modules on a device. Each ePWM module has a synchronization input (EPWMxSYNCI) and a synchronization output (EPWMxSYNCO). The input synchronization for the first instance (ePWM1) comes from an external pin.

Figure 7-7. Time-Base Counter Synchronization Scheme 4



NOTE: All modules shown in the synchronization schemes may not be available on all devices. Please refer to the device specific data manual to determine which modules are available on a particular device.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: Time-Base Counter Synchronization Scheme 4 is specifically applicable to ePWM type 2. On this device, If a pin will be used as a C28 GPIO, the correct bits must be set in the GPIOCSEL register. This register is located in the M3 GPIO register space and more details can be found in the General-Purpose Input/Output (GPIO) chapter of this TRM. Once the user configures the respective pin as a C28 GPIO then the GPTRIP6SEL register which belongs to the GPTRIP6 logic should be used to configure that specific pin as an external SYNC input. The user is responsible for driving low state on the selected pin before enabling clock for the respective ePWM peripheral to avoid spurious latch of SYNC signal

Each ePWM module can be configured to use or ignore the synchronization input. If the TBCTL[PHSEN] bit is set, then the time-base counter (TBCTR) of the ePWM module will be automatically loaded with the phase register (TBPHS) contents when one of the following conditions occur:

- **EPWMxSYNCI: Synchronization Input Pulse:**

  The value of the phase register is loaded into the counter register when an input synchronization pulse is detected (TBPHS → TBCTR). This operation occurs on the next valid time-base clock (TBCLK) edge.

  The delay from internal master module to slave modules is given by:

  – if (TBCLK = SYSCLKOUT): 2 x SYSCLKOUT
  – if (TBCLK != SYSCLKOUT):1 TBCLK

- **Software Forced Synchronization Pulse:**

  Writing a 1 to the TBCTL[SWFSYNC] control bit invokes a software forced synchronization. This pulse is ORed with the synchronization input signal, and therefore has the same effect as a pulse on EPWMxSYNCI.

- **Digital Compare Event Synchronization Pulse:**

  DCAEVT1 and DCBEVT1 digital compare events can be configured to generate synchronization pulses which have the same affect as EPWMxSYNCI.

This feature enables the ePWM module to be automatically synchronized to the time base of another ePWM module. Lead or lag phase control can be added to the waveforms generated by different ePWM modules to synchronize them. In up-down-count mode, the TBCTL[PSHDIR] bit configures the direction of the time-base counter immediately after a synchronization event. The new direction is independent of the direction prior to the synchronization event. The PHSDIR bit is ignored in count-up or count-down modes. See Figure 7-8 through Figure 7-11 for examples.

Clearing the TBCTL[PHSEN] bit configures the ePWM to ignore the synchronization input pulse. The synchronization pulse can still be allowed to flow-through to the EPWMxSYNCO and be used to synchronize other ePWM modules. In this way, you can set up a master time-base (for example, ePWM1) and downstream modules (ePWM2 - ePWMx) may elect to run in synchronization with the master. See the Application to Power Topologies Section 7.3 for more details on synchronization strategies.

7.2.2.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules

The TBCLKSYNC bit can be used to globally synchronize the time-base clocks of all enabled ePWM modules on a device. This bit is part of the device's clock enable registers and is described in the *System Control and Interrupts* section of this manual. When TBCLKSYNC = 0, the time-base clock of all ePWM modules is stopped (default). When TBCLKSYNC = 1, all ePWM time-base clocks are started with the rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the TBCTL register of each ePWM module must be set identically. The proper procedure for enabling the ePWM clocks is as follows:

1. Enable the individual ePWM module clocks. This is described in the device-specific version of the *System Control and Interrupts Reference Guide*.
2. Set TBCLKSYNC = 0. This will stop the time-base clock within any enabled ePWM module.
3. Configure the prescaler values and desired ePWM modes.
4. Set TBCLKSYNC = 1.

Copyright © 2012–2019, Texas Instruments Incorporated

7.2.2.5 Simultaneous Writes to TBPRD and CMPx Registers Between ePWM Modules

For variable frequency applications, there is a need for simultaneous writes of TBPRD and CMPx registers between ePWM modules. This prevents situations where a CTR = 0 or CTR = PRD pulse forces a shadow to active load of these registers before all registers are updated between ePWM modules (resulting in some registers being loaded from new shadow values while others are loaded from old shadow values). To support this, an ePWM register linking scheme for TBPRD:TBPRDHR, CMPA:CMPAHR, CMPB:CMPBHR, CMPC, and CMPD registers between PWM modules has been added.

For a particular ePWM module # A , user code writes “B+1”, to the linked register bit-field in EPWMXLINK. “B” is the ePWM module # being linked to (i.e. Writes to the ePWM module “B” TBPRD:TBPRDHR, CMPA:CMPAHR, CMPB:CMPBHR, or CMPC will simultaneously be written to corresponding register in ePWM module “A”) For instance if ePWM3 EPWMXLINK register is configured so that CMPA:CMPAHR are linked to ePWM1, then a write to CMPA:CMPAHR in ePWM 1 will simultaneously write the same value to CMPA:CMPAHR in ePWM3. If ePWM4 also has its CMPA:CMPAHR registers linked to ePWM1, then a write to ePWM 1 will write the same value to the CMPA:CMPAHR registers in both ePWM3 and ePWM4.

The register description for EPWMXLINK clearly explains the linked register bit-field values for corresponding ePWM.

NOTE: ePWM register linking scheme works with the TBPRD:TBPRDHR [Mirrored instance], CMPA:CMPAHR [Mirrored instance], CMPB:CMPBHR [Mirrored instance], CMPC, and CMPD registers present in the upper page offset. Only the instance of these registers in offsets 0x62-0x6B are linked between ePWM modules

A typical example snippet will use the following registers linked between modules

EPwmxRegs.TBPRDHRM2 , EPwmxRegs.CMPAM2 , EPwmxRegs.CMPBM , EPwmxRegs.CMPC , EPwmxRegs.CMPD

7.2.2.6 Time-base Counter Modes and Timing Waveforms

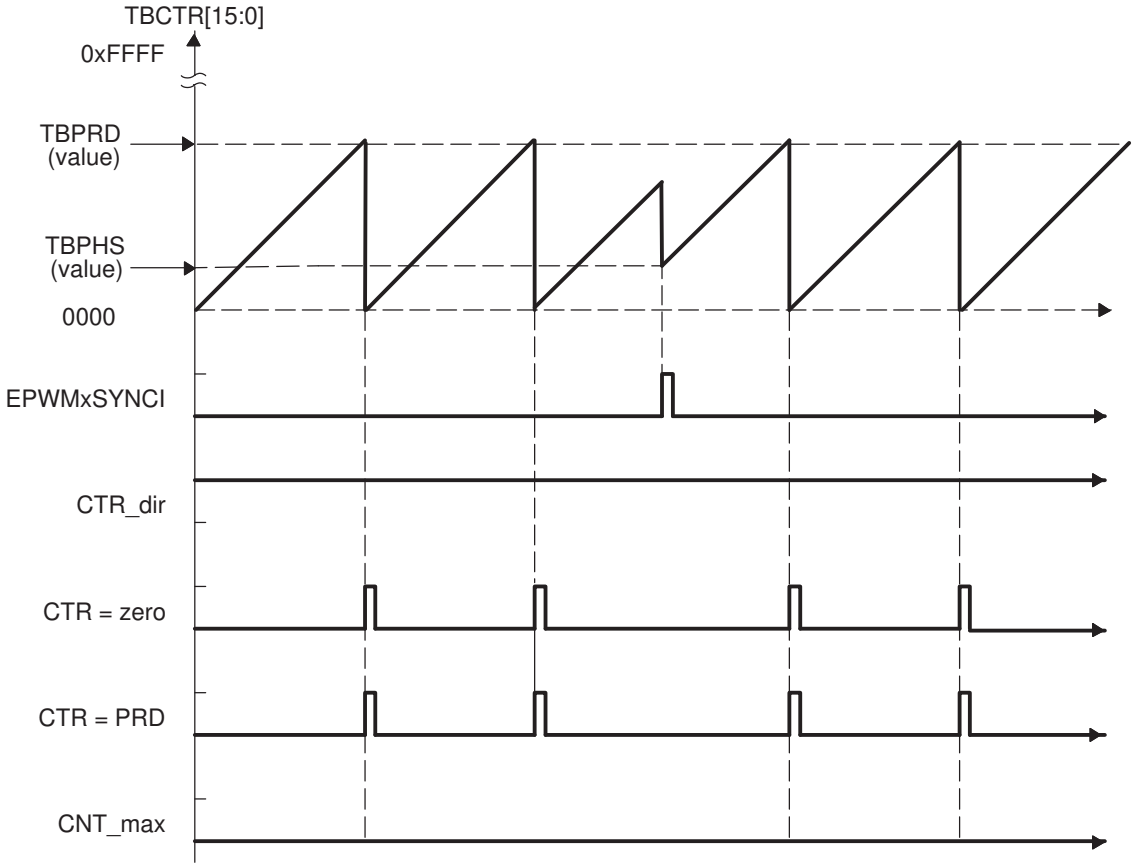
The time-base counter operates in one of four modes:

- Up-count mode which is asymmetrical.
- Down-count mode which is asymmetrical.
- Up-down-count which is symmetrical
- Frozen where the time-base counter is held constant at the current value

To illustrate the operation of the first three modes, the following timing diagrams show when events are generated and how the time-base responds to an EPWMxSYNCI signal.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-8. Time-Base Up-Count Mode Waveforms



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-9. Time-Base Down-Count Mode Waveforms

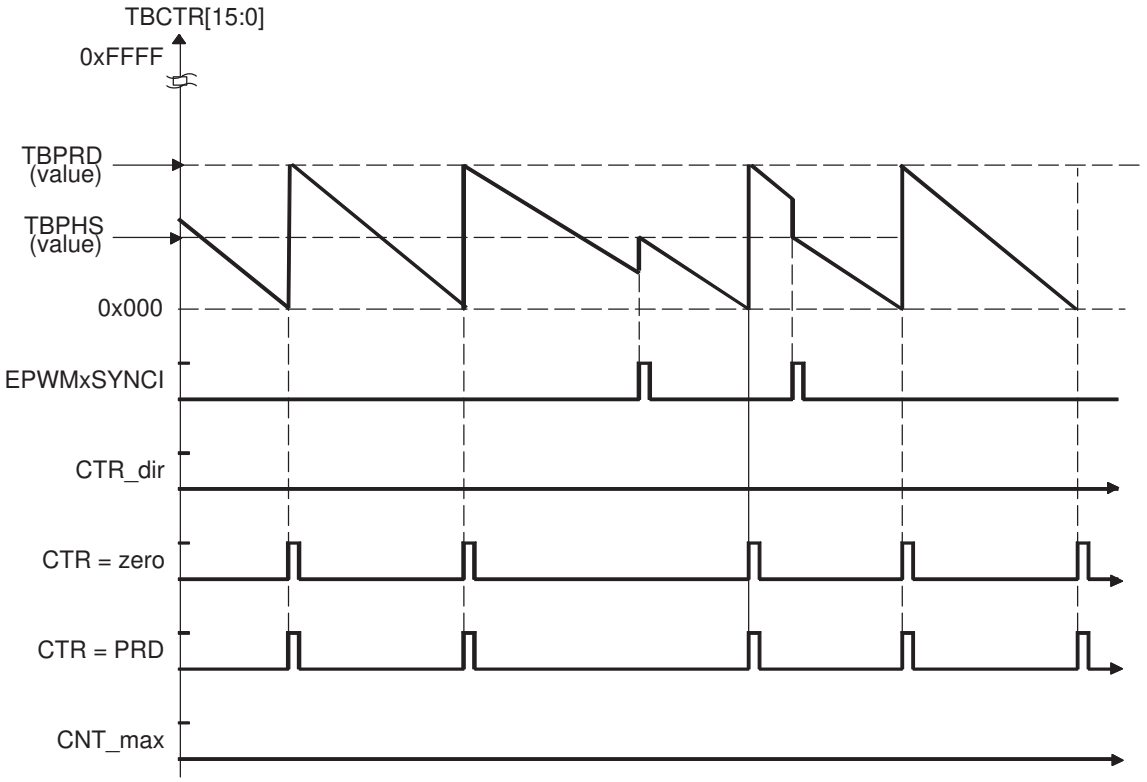
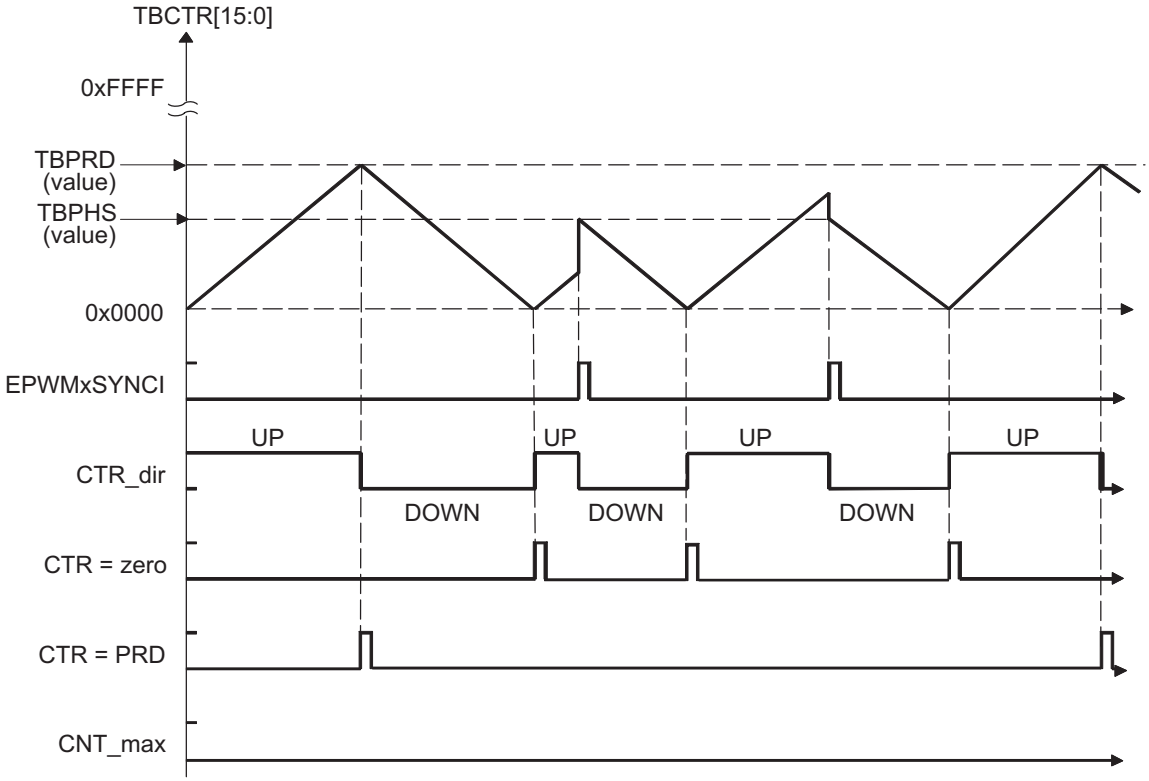
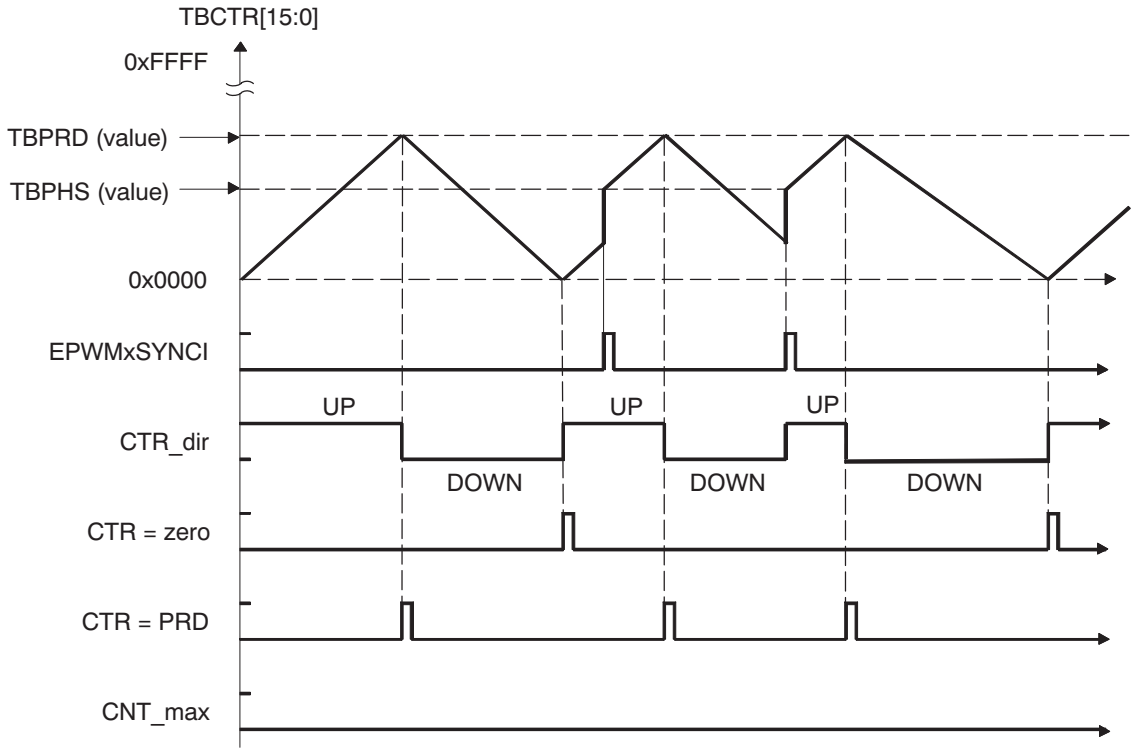


Figure 7-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event



7.2.3 Counter-Compare (CC) Submodule

Figure 7-12 illustrates the counter-compare submodule within the ePWM.

Figure 7-12. Counter-Compare Submodule

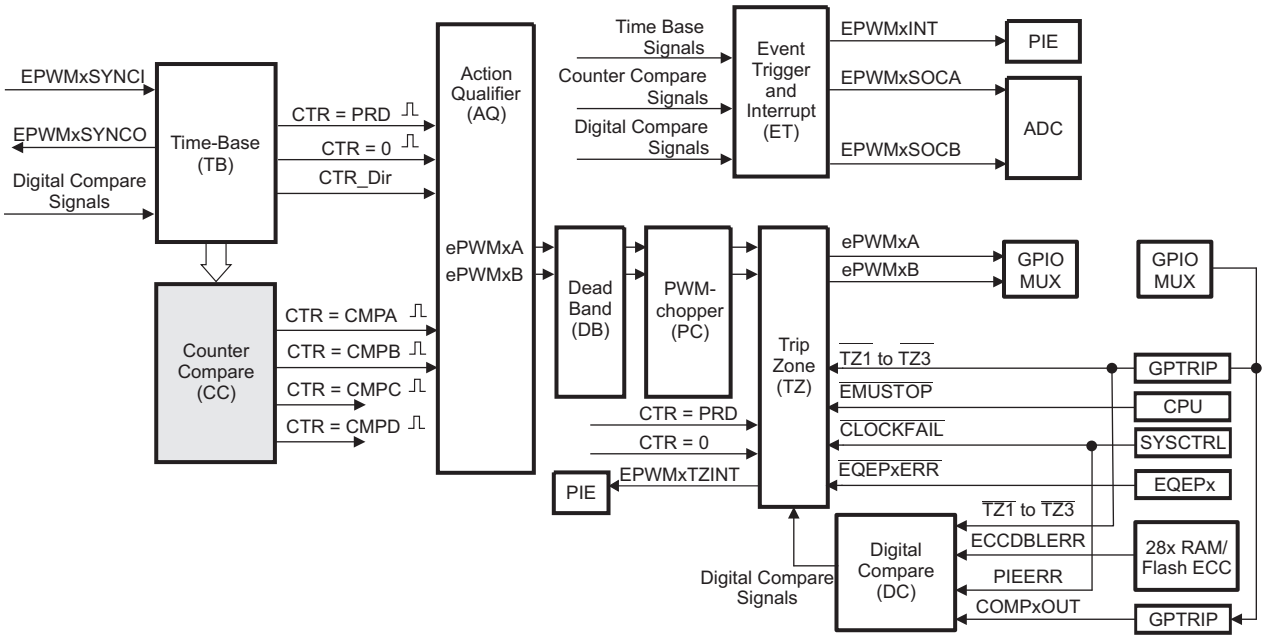


Figure 7-13 shows the basic structure of the counter-compare submodule.

Copyright © 2012–2019, Texas Instruments Incorporated

7.2.3.1 Purpose of the Counter-Compare Submodule

The counter-compare submodule takes as input the time-base counter value. This value is continuously compared to the counter-compare A (CMPA) counter-compare B (CMPB) counter-compare C (CMPC) and counter-compare D (CMPD)registers. When the time-base counter is equal to one of the compare registers, the counter-compare unit generates an appropriate event.

The counter-compare:

- Generates events based on programmable time stamps using the CMPA, CMPB, CMPC and CMPD registers
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCTR = CMPA).
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCTR = CMPB)
  - CTR = CMPC: Time-base counter equals counter-compare C register (TBCTR = CMPC).
  - CTR = CMPD: Time-base counter equals counter-compare D register (TBCTR = CMPD)
- Controls the PWM duty cycle if the action-qualifier submodule is configured appropriately using counter-compare A (CMPA) & counter-compare B (CMPB)
- Shadows new compare values to prevent corruption or glitches during the active PWM cycle

7.2.3.2 Controlling and Monitoring the Counter-Compare Submodule

The counter-compare submodule operation is controlled and monitored by the registers shown in Table 7-6:

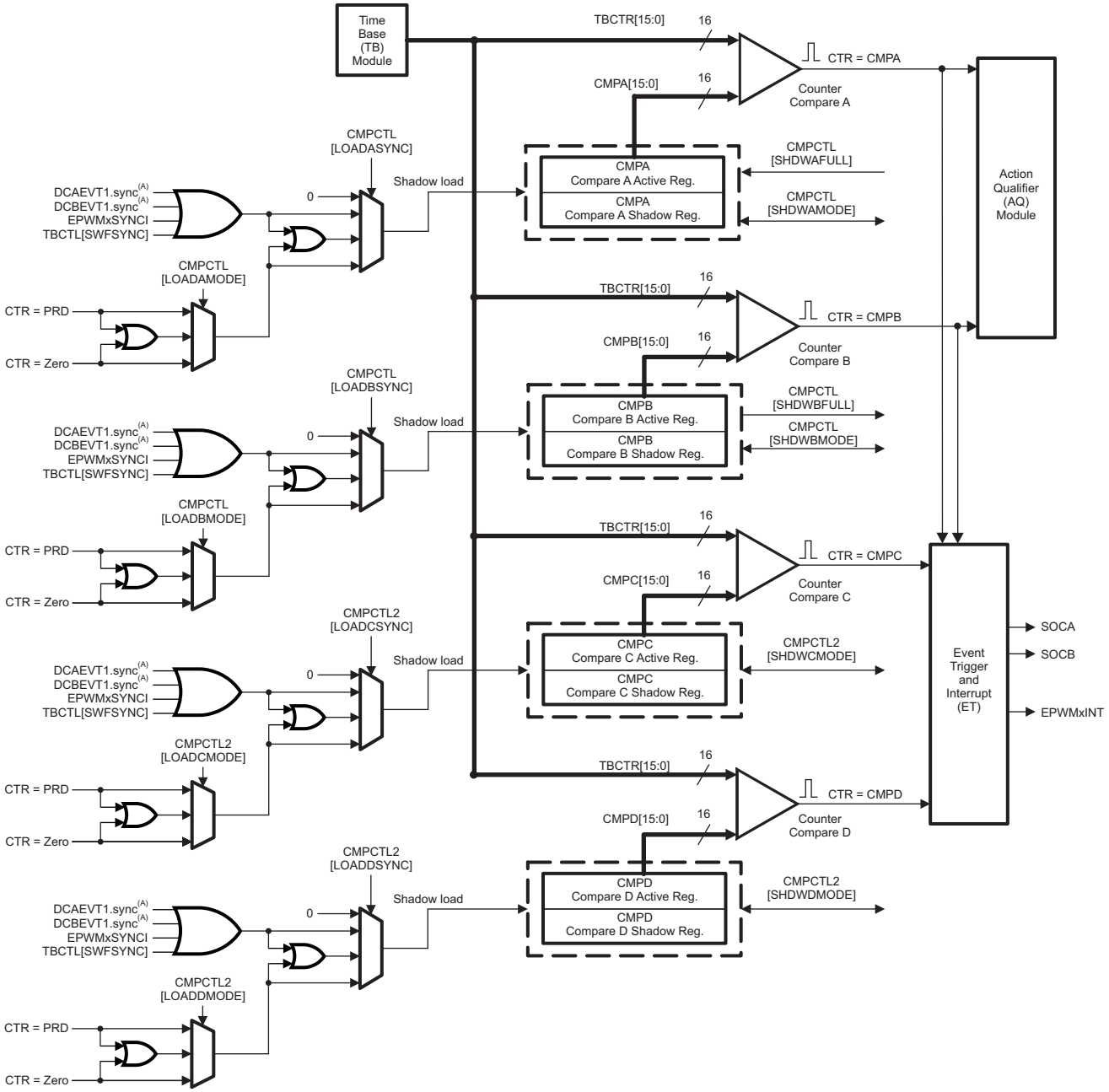
Table 7-6. Counter-Compare Submodule Registers

Register Name	Address Offset	Shadowed	Description
CMPCTL	0x07	No	Counter-Compare Control Register
CMPCTL2	0x41	No	Counter Compare Control Register 2
CMPAHR	0x08	Yes	HRPWM Counter-Compare A Extension Register(1)
CMPA	0x09	Yes	Counter-Compare A Register
CMPAHRM	0x2C	Writes	HRPWM Counter-Compare A Extension Mirror Register(1)
CMPAM	0x2D	Writes	Counter-Compare A Mirror Register
CMPAHRM2	0x64	Yes	Counter Compare A High-Resolution Mirror 2 Register(1)
CMPAM2	0x65	Yes	Counter Compare A Mirror 2 Register(1)
CMPB	0x0A	Yes	Counter-Compare B Register
CMPBHR	0x4A	Yes	Compare B High-Resolution Register
CMPBHRM	0x66	Writes	Counter Compare B High-Resolution Mirror Register(1)
CMPBM	0x67	Writes	Counter Compare B Mirror Register
CMPC	0x69	Yes	Counter Compare C Register
CMPD	0x6B	Yes	Counter Compare D Register

(1) This register is available only on ePWM modules with the high-resolution extension (HRPWM). On ePWM modules that do not include the HRPWM this location is reserved. This register is described in the device-specific *High-Resolution Pulse Width Modulator (HRPWM)* section of this manual. Refer to the device-specific data manual to determine which ePWM instances include this feature.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-13. Detailed View of the Counter-Compare Submodule



A These events are generated by the type 2 ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs [For Example: COMPxOUT and $\overline{TZ}$ signals].

The key signals associated with the counter-compare submodule are described in Table 7-7.

Table 7-7. Counter-Compare Submodule Key Signals

Signal	Description of Event	Registers Compared
CTR = CMPA	Time-base counter equal to the active counter-compare A value	TBCTR = CMPA
CTR = CMPB	Time-base counter equal to the active counter-compare B value	TBCTR = CMPB
CTR = CMPC	Time-base counter equal to the active counter-compare C value	TBCTR = CMPC
CTR = CMPD	Time-base counter equal to the active counter-compare D value	TBCTR = CMPD

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-7. Counter-Compare Submodule Key Signals (continued)

Signal	Description of Event	Registers Compared
CTR = PRD	Time-base counter equal to the active period Used to load active counter-compare A and B registers from the shadow register.	TBCTR = TBPRD
CTR = ZERO	Time-base counter equal to zero Used to load active counter-compare A and B registers from the shadow register.	TBCTR = 0x00

7.2.3.3 Operational Highlights for the Counter-Compare Submodule

The counter-compare submodule is responsible for generating two independent compare events based on two compare registers which is fed to action-qualifier submodule and Event Trigger submodule :

1. CTR = CMPA: Time-base counter equal to counter-compare A register (TBCTR = CMPA).
2. CTR = CMPB: Time-base counter equal to counter-compare B register (TBCTR = CMPB).

For up-count or down-count mode, each event occurs only once per cycle. For up-down-count mode each event occurs twice per cycle if the compare value is between 0x00-TBPRD and once per cycle if the compare value is equal to 0x00 or equal to TBPRD. These events are fed into the action-qualifier submodule where they are qualified by the counter direction and converted into actions if enabled. Refer to Section 7.2.4.1 for more details.

The counter-compare registers CMPA and CMPB each have an associated shadow register. Shadowing provides a way to keep updates to the registers synchronized with the hardware. When shadowing is used, updates to the active registers only occur at strategic points. This prevents corruption or spurious operation due to the register being asynchronously modified by software. The memory address of the active register and the shadow register is identical. Which register is written to or read from is determined by the CMPCTL[SHDWAMODE] and CMPCTL[SHDWBMODE] bits. These bits enable and disable the CMPA shadow register and CMPB shadow register respectively. The behavior of the two load modes is described below:

Shadow Mode:

The shadow mode for the CMPA is enabled by clearing the CMPCTL[SHDWAMODE] bit and the shadow register for CMPB is enabled by clearing the CMPCTL[SHDWBMODE] bit. Shadow mode is enabled by default for both CMPA and CMPB.

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events as specified by the CMPCTL[LOADAMODE] CMPCTL[LOADBMODE] CMPCTL[LOADASYNC] & CMPCTL[LOADBSYNC] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero
- SYNC event caused by DCAEVT1 or DCBEVT1 or EPWMxSYNCI or TBCTL[SWFSYNC]
- Both SYNC event or a selection made by LOADAMODE/LOADBMODE

Only the active register contents are used by the counter-compare submodule to generate events to be sent to the action-qualifier.

Immediate Mode:

If immediate load mode is selected (i.e., CMPCTL[SHDWAMODE] = 1 or CMPCTL[SHDWBMODE] = 1), then a read from or a write to the register will go directly to the active register.

Addtional Comparators on ePWM type 2

The counter-compare submodule on ePWM type 2 is responsible for generating two additional independent compare events based on two compare registers which is fed to Event Trigger submodule :

1. CTR = CMPC: Time-base counter equal to counter-compare C register (TBCTR = CMPC).
2. CTR = CMPD: Time-base counter equal to counter-compare D register (TBCTR = CMPD).

Copyright © 2012–2019, Texas Instruments Incorporated

The counter-compare registers CMPC and CMPD each have an associated shadow register. By default this register is shadowed. The memory address of the active register and the shadow register is identical.The value in the active CMPC & CMPD register is compared to the time-base counter (TBCTR). When the values are equal, the counter compare module generates a “time-base counter equal to counter compare C or counter compare D ” event respectively. Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWCMODE] and CMPCTL2[SHDWDMODE] bit. These bits enable and disable the CMPA shadow register and CMPB shadow register respectively. The behavior of the two load modes is described below:

Shadow Mode:

The shadow mode for the CMPC is enabled by clearing the CMPCTL2[SHDWCMODE] bit and the shadow register for CMPD is enabled by clearing the CMPCTL2[SHDWDMODE] bit. Shadow mode is enabled by default for both CMPC and CMPD.

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events as specified by the CMPCTL2[LOADCMODE] CMPCTL2[LOADDMODE] CMPCTL2[LOADCSYNC] & CMPCTL2[LOADDSYNC] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero
- SYNC event caused by DCAEVT1 or DCBEVT1 or EPWMxSYNCI or TBCTL[SWFSYNC]
- Both SYNC event or a selection made by LOADCMODE/LOADDMODE

Only the active register contents are used by the counter-compare submodule to generate events to be sent to the action-qualifier.

Immediate Load Mode:

If immediate load mode is selected (i.e., CMPCTL2[SHDWCMODE] = 1 or CMPCTL2[SHDWDMODE] = 1), then a read from or a write to the register will go directly to the active register.

7.2.3.4 Count Mode Timing Waveforms

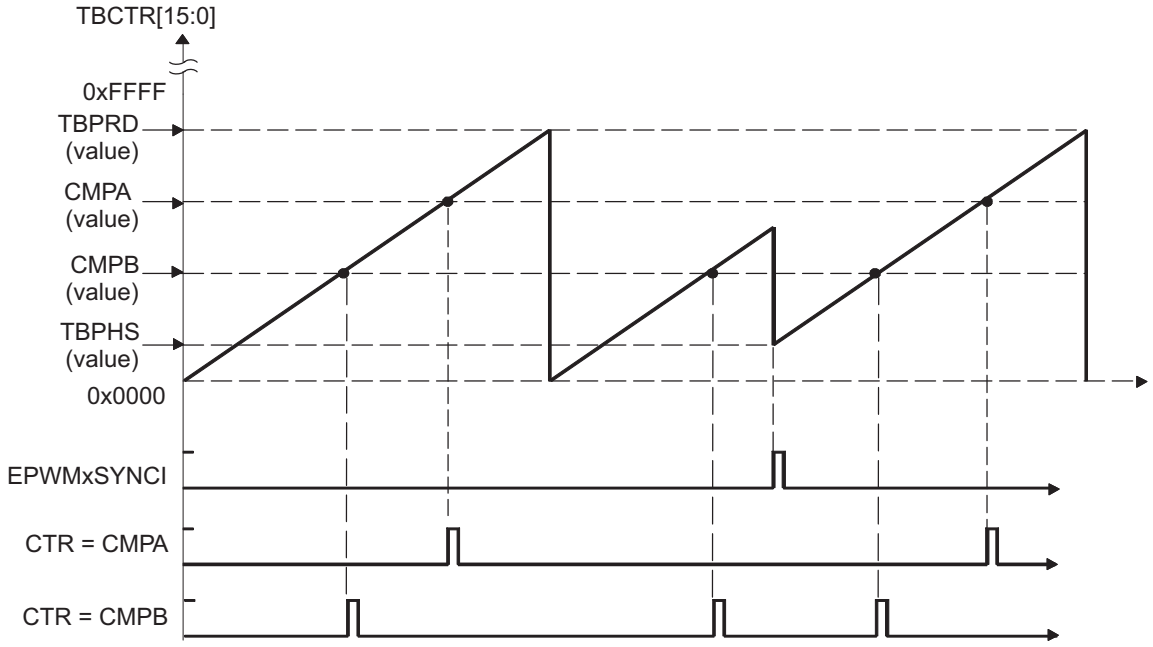
The counter-compare module can generate compare events in all three count modes:

- Up-count mode: used to generate an asymmetrical PWM waveform.
- Down-count mode: used to generate an asymmetrical PWM waveform.
- Up-down-count mode: used to generate a symmetrical PWM waveform.

To best illustrate the operation of the first three modes, the timing diagrams in Figure 7-14 through Figure 7-17 show when events are generated and how the EPWMxSYNCI signal interacts.

Copyright © 2012–2019, Texas Instruments Incorporated

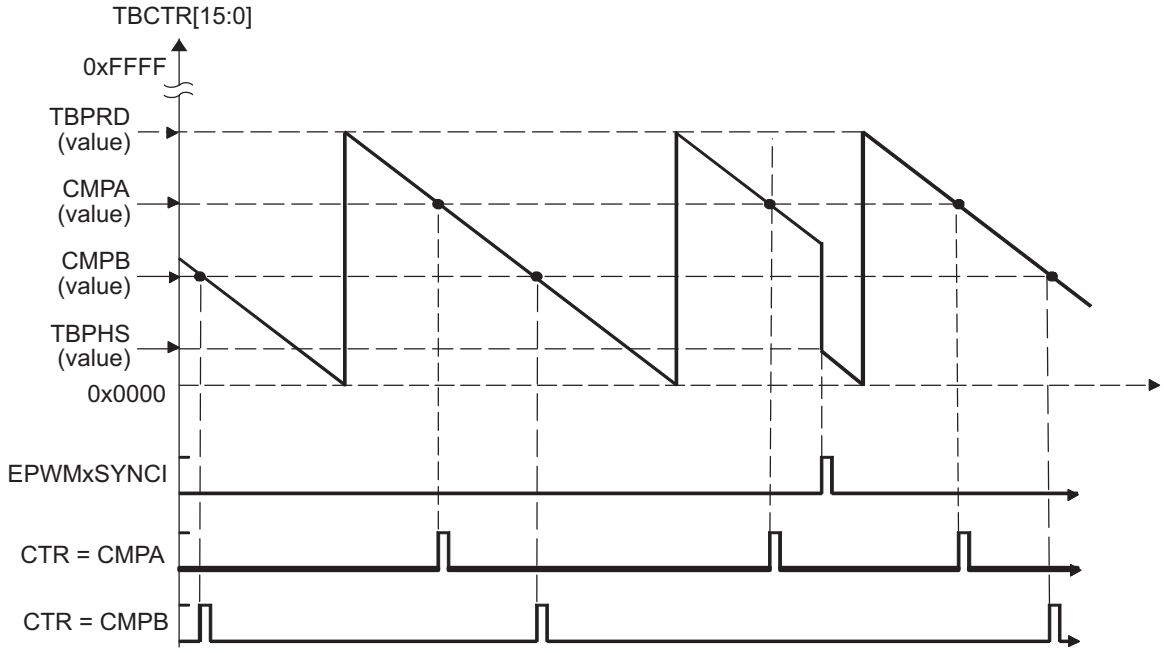
Figure 7-14. Counter-Compare Event Waveforms in Up-Count Mode



NOTE: An EPWMxSYNCI external synchronization event can cause a discontinuity in the TBCTR count sequence. This can lead to a compare event being skipped. This skipping is considered normal operation and must be taken into account.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-15. Counter-Compare Events in Down-Count Mode



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-16. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event

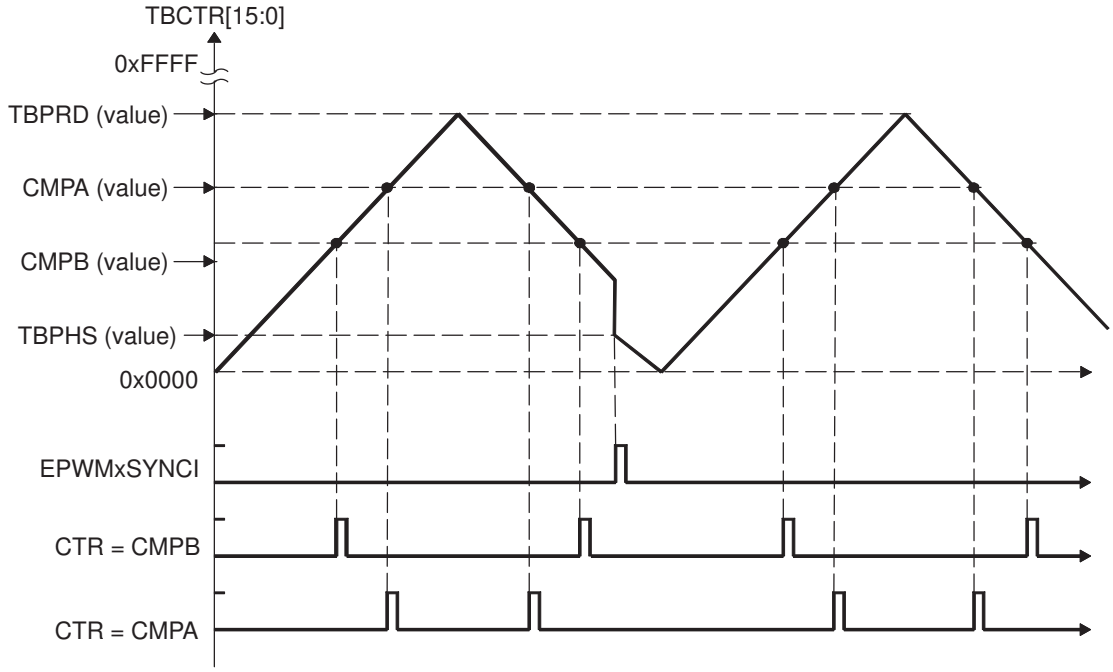
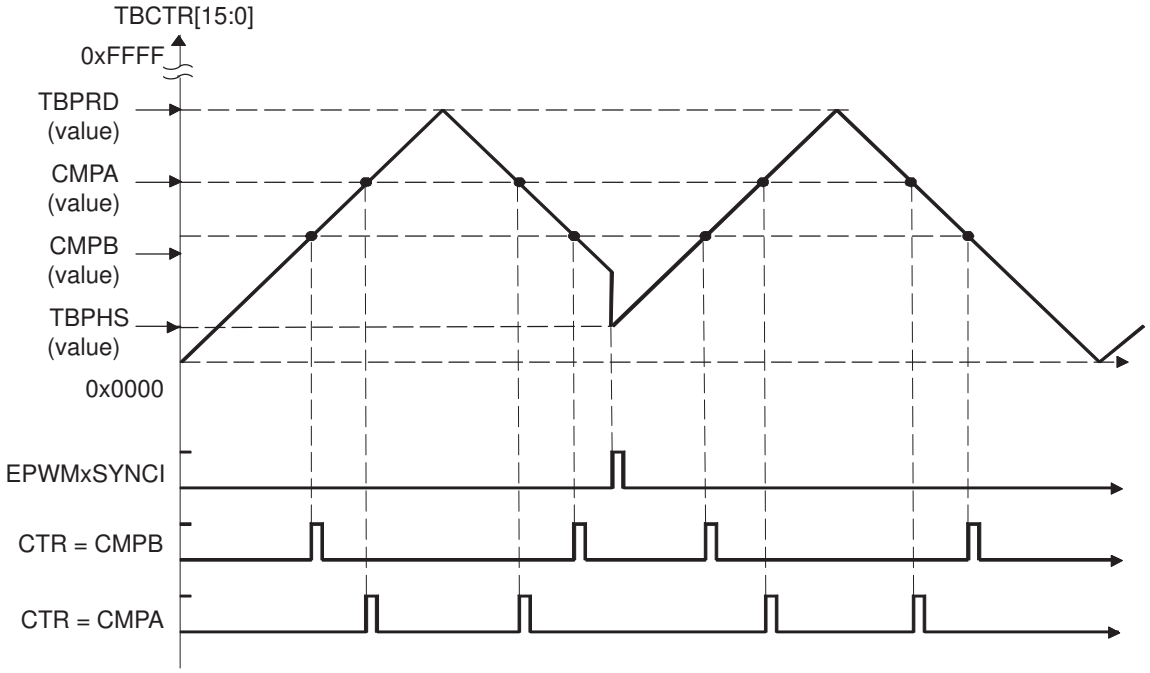


Figure 7-17. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event

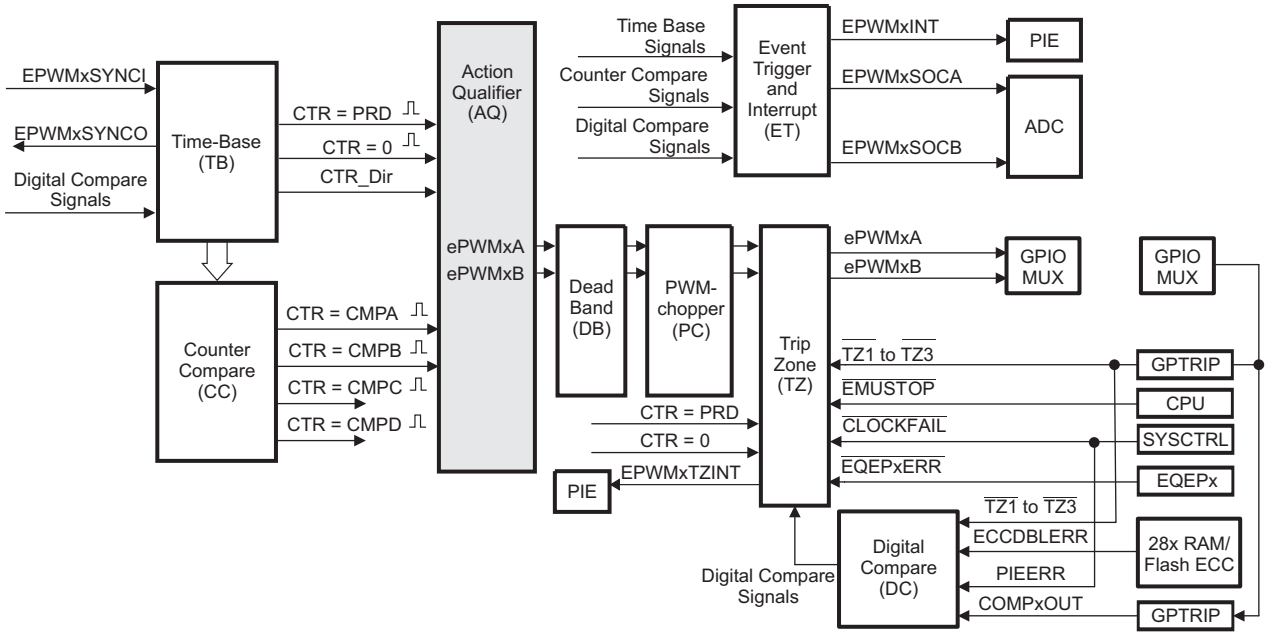


7.2.4 Action-Qualifier (AQ) Submodule

Figure 7-18 shows the action-qualifier (AQ) submodule (see shaded block) in the ePWM system.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-18. Action-Qualifier Submodule



The action-qualifier submodule has the most important role in waveform construction and PWM generation. It decides which events are converted into various action types, thereby producing the required switched waveforms at the EPWMxA and EPWMxB outputs.

7.2.4.1 Purpose of the Action-Qualifier Submodule

The action-qualifier submodule is responsible for the following:

- Qualifying and generating actions (set, clear, toggle) based on the following events:
  - CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
  - CTR = CMPA: Time-base counter equal to the counter-compare A register (TBCTR = CMPA)
  - CTR = CMPB: Time-base counter equal to the counter-compare B register (TBCTR = CMPB)
- Managing priority when these events occur concurrently
- Providing independent control of events when the time-base counter is increasing and when it is decreasing

7.2.4.2 Action-Qualifier Submodule Control and Status Register Definitions

The action-qualifier submodule operation is controlled and monitored via the registers in Table 7-8.

Table 7-8. Action-Qualifier Submodule Registers

Register Name	Address Offset	Shadowed	Description
AQCTLA	0x0B	No	Action-Qualifier Control Register For Output A (EPWMxA)
AQCTLB	0x0C	No	Action-Qualifier Control Register For Output B (EPWMxB)
AQSFRC	0x0D	No	Action-Qualifier Software Force Register
AQCSFRC	0x0E	Yes	Action-Qualifier Continuous Software Force
AQCTLR	0x47	No	Action Qualifier Control Register
AQCTLAM	0x73	No	Action Qualifier Control Mirror Register For Output A
AQCTLBM	0x74	No	Action Qualifier Control Mirror Register For Output B
AQSFRCM	0x75	No	Action Qualifier Software Force Mirror Register

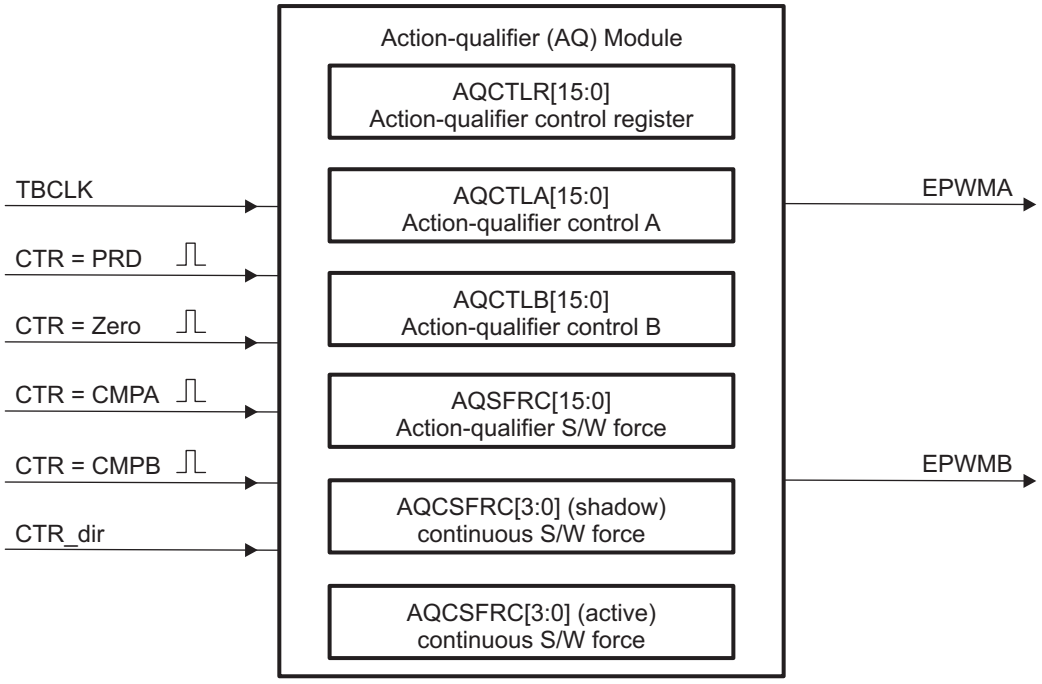
Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-8. Action-Qualifier Submodule Registers (continued)

Register Name	Address Offset	Shadowed	Description
AQCSFRCM	0x76	No	Action Qualifier Continuous S/W Force Mirror Register
AQCTLR	0x47	No	Action Qualifier Control Register

The action-qualifier submodule is based on event-driven logic. It can be thought of as a programmable cross switch with events at the input and actions at the output, all of which are software controlled via the set of registers shown in Table 7-8.

Figure 7-19. Action-Qualifier Submodule Inputs and Outputs



For convenience, the possible input events are summarized again in Table 7-9.

Table 7-9. Action-Qualifier Submodule Possible Input Events

Signal	Description	Registers Compared
CTR = PRD	Time-base counter equal to the period value	TBCTR = TBPRD
CTR = Zero	Time-base counter equal to zero	TBCTR = 0x00
CTR = CMPA	Time-base counter equal to the counter-compare A	TBCTR = CMPA
CTR = CMPB	Time-base counter equal to the counter-compare B	TBCTR = CMPB
Software forced event	Asynchronous event initiated by software	

The software forced action is a useful asynchronous event. This control is handled by registers AQSFRC and AQCSFRC.

The action-qualifier submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs. The event inputs to the action-qualifier submodule are further qualified by the counter direction (up or down). This allows for independent action on outputs on both the count-up and count-down phases.

The possible actions imposed on outputs EPWMxA and EPWMxB are:

- **Set High:**

  Set output EPWMxA or EPWMxB to a high level.

Copyright © 2012–2019, Texas Instruments Incorporated

- **Clear Low:**

  Set output EPWMxA or EPWMxB to a low level.

- **Toggle:**

  If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.

- **Do Nothing:**

  Keep outputs EPWMxA and EPWMxB at same level as currently set. Although the "Do Nothing" option prevents an event from causing an action on the EPWMxA and EPWMxB outputs, this event can still trigger interrupts and ADC start of conversion. See the Event-trigger Submodule description in Section 7.2.8 for details.

Actions are specified independently for either output (EPWMxA or EPWMxB). Any or all events can be configured to generate actions on a given output. For example, both CTR = CMPA and CTR = CMPB can operate on output EPWMxA. All qualifier actions are configured via the control registers found at the end of this section.

For clarity, the drawings in this document use a set of symbolic actions. These symbols are summarized in Figure 7-20. Each symbol represents an action as a marker in time. Some actions are fixed in time (zero and period) while the CMPA and CMPB actions are moveable and their time positions are programmed via the counter-compare A and B registers, respectively. To turn off or disable an action, use the "Do Nothing option"; it is the default at reset.

Figure 7-20. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs

S/W force	TB Counter equals:				Actions
	Zero	Comp A	Comp B	Period	
SW ✕	Z ✕	CA ✕	CB ✕	P ✕	Do Nothing
SW ↓	Z ↓	CA ↓	CB ↓	P ↓	Clear Low
SW ↑	Z ↑	CA ↑	CB ↑	P ↑	Set High
SW T	Z T	CA T	CB T	P T	Toggle

Copyright © 2012–2019, Texas Instruments Incorporated

7.2.4.3 Action-Qualifier Event Priority

It is possible for the ePWM action qualifier to receive more than one event at the same time. In this case events are assigned a priority by the hardware. The general rule is events occurring later in time have a higher priority and software forced events always have the highest priority. The event priority levels for up-down-count mode are shown in Table 7-10. A priority level of 1 is the highest priority and level 7 is the lowest. The priority changes slightly depending on the direction of TBCTR.

Table 7-10. Action-Qualifier Event Priority for Up-Down-Count Mode

Priority Level	Event If TBCTR is Incrementing TBCTR = Zero up to TBCTR = TBPRD	Event If TBCTR is Decrementing TBCTR = TBPRD down to TBCTR = 1
1 (Highest)	Software forced event	Software forced event
2	Counter equals CMPB on up-count (CBU)	Counter equals CMPB on down-count (CBD)
3	Counter equals CMPA on up-count (CAU)	Counter equals CMPA on down-count (CAD)
4	Counter equals zero	Counter equals period (TBPRD)
5	Counter equals CMPB on down-count (CBD)	Counter equals CMPB on up-count (CBU)
6 (Lowest)	Counter equals CMPA on down-count (CAD)	Counter equals CMPA on up-count (CBU)

Table 7-11 shows the action-qualifier priority for up-count mode. In this case, the counter direction is always defined as up and thus down-count events will never be taken.

Table 7-11. Action-Qualifier Event Priority for Up-Count Mode

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to period (TBPRD)
3	Counter equal to CMPB on up-count (CBU)
4	Counter equal to CMPA on up-count (CAU)
5 (Lowest)	Counter equal to Zero

Table 7-12 shows the action-qualifier priority for down-count mode. In this case, the counter direction is always defined as down and thus up-count events will never be taken.

Table 7-12. Action-Qualifier Event Priority for Down-Count Mode

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to Zero
3	Counter equal to CMPB on down-count (CBD)
4	Counter equal to CMPA on down-count (CAD)
5 (Lowest)	Counter equal to period (TBPRD)

It is possible to set the compare value greater than the period. In this case the action will take place as shown in Table 7-13.

Table 7-13. Behavior if CMPA/CMPB is Greater than the Period

Counter Mode	Compare on Up-Count Event CAD/CBD	Compare on Down-Count Event CAD/CBD
Up-Count Mode	If CMPA/CMPB ≤ TBPRD period, then the event occurs on a compare match (TBCTR=CMPA or CMPB). If CMPA/CMPB > TBPRD, then the event will not occur.	Never occurs.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-13. Behavior if CMPA/CMPB is Greater than the Period (continued)

Counter Mode	Compare on Up-Count Event CAD/CBD	Compare on Down-Count Event CAD/CBD
Down-Count Mode	Never occurs.	If CMPA/CMPB < TBPRD, the event will occur on a compare match (TBCTR=CMPA or CMPB).
		If CMPA/CMPB ≥ TBPRD, the event will occur on a period match (TBCTR=TBPRD).
Up-Down-Count Mode	If CMPA/CMPB < TBPRD and the counter is incrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB).	If CMPA/CMPB < TBPRD and the counter is decrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB).
	If CMPA/CMPB is ≥ TBPRD, the event will occur on a period match (TBCTR = TBPRD).	If CMPA/CMPB ≥ TBPRD, the event occurs on a period match (TBCTR=TBPRD).

7.2.4.4 AQCTLA and AQCTLB shadow mode operations

To enable Action Qualifier mode changes which must occur at the end of a period even when the phase changes, shadowing of the AQCTLA and AQCTLB registers has been added on ePWM type 2. Additionally, shadow to active load on SYNC of these registers is supported as well. Shadowing of this register is enabled and disabled by the AQCTLR[SHDWAQAMODE] and AQCTLR[SHDWAQBMODE] bits. These bits enable and disable the AQCTLA shadow register and AQCTLB shadow register, respectively. The behavior of the two load modes is described below:

Shadow Mode:

The shadow mode for the AQCTLA is enabled by setting the AQCTLR[SHDWAQAMODE] bit, and the shadow register for AQCTLB is enabled by setting the AQCTLR[SHDWAQBMODE] bit. Shadow mode is disabled by default for both AQCTLA and AQCTLB

If the shadow register is enabled, then the content of the shadow register is transferred to the active register on one of the following events as specified by the AQCTLR[LDAQAMODE] AQCTLR[LDAQBMODE] AQCTLR[LDAQASYNC] & AQCTLR[LDAQBSYNC] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero
- SYNC event caused by DCAEVT1 or DCBEVT1 or EPWMxSYNCI or TBCTL[SWFSYNC]
- Both SYNC event or a selection made by LDAQAMODE/LDAQBMODE

Immediate Load Mode:

If immediate load mode is selected (i.e., AQCTLR[SHDWAQAMODE] = 0 or AQCTLR[SHDWAQBMODE] = 0), then a read from or a write to the register will go directly to the active register. See Figure 7-21 and Figure 7-22.

NOTE: Shadow to Active Load of Action Qualifier Output A/B Control Register [AQCTLA & AQCTLB] on CMPA = 0 or CMPB = 0 boundary

If the Counter-Compare A Register (CMPA) or Counter-Compare B Register (CMPB) is set to a value of 0 and the action qualifier action on AQCTLA and AQCTLB is configured to occur in the same instant as a shadow to active load (i.e., CMPA=0 and AQCTLA shadow to active load on TBCTR=0 using AQCTLR register LDAQAMODE and LDAQAMODE bits), then both events enter contention and it is recommended to use a Non-Zero Counter-Compare when using Shadow to Active Load of Action Qualifier Output A/B Control Register on TBCTR = 0 boundary.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-21. AQCTLR[SHDWAQAMODE]

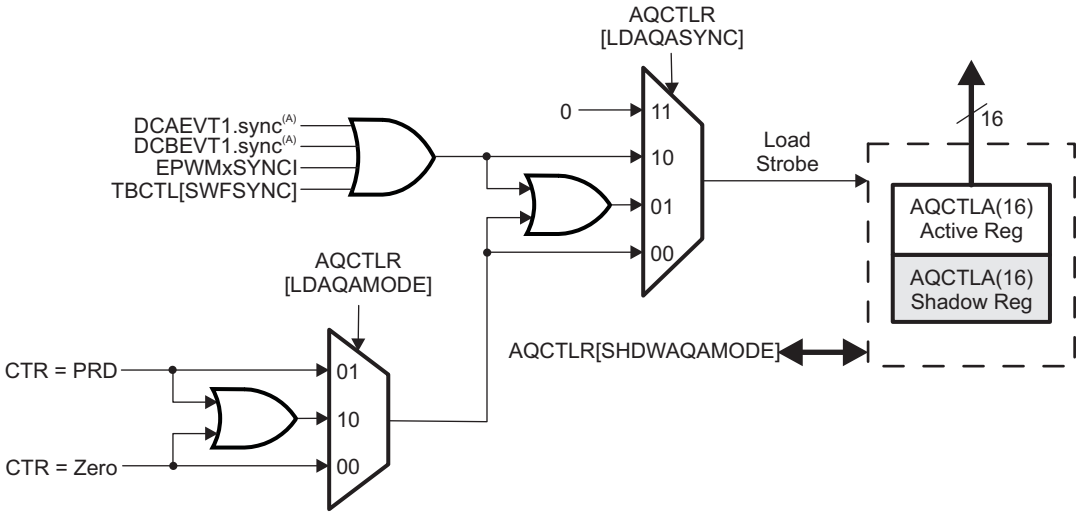
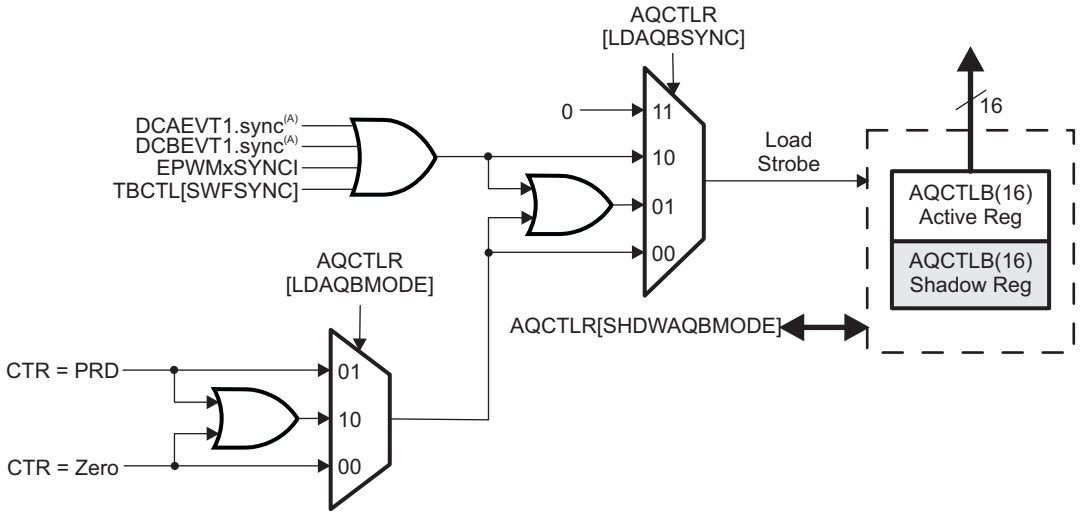


Figure 7-22. AQCTLR[SHDWAQBMODE]



Copyright © 2012–2019, Texas Instruments Incorporated

7.2.4.5 Waveforms for Common Configurations

NOTE: The waveforms in this document show the ePWMs behavior for a static compare register value. In a running system, the active compare registers (CMPA and CMPB) are typically updated from their respective shadow registers once every period. The user specifies when the update will take place; either when the time-base counter reaches zero or when the time-base counter reaches period. There are some cases when the action based on the new value can be delayed by one period or the action based on the old value can take effect for an extra period. Some PWM configurations avoid this situation. These include, but are not limited to, the following:

Use up-down-count mode to generate a symmetric PWM:

- If you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1.
- If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1.

  This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

Use up-down-count mode to generate an asymmetric PWM:

- To achieve 50%-0% asymmetric PWM use the following configuration: Load CMPA/CMPB on period and use the period action to clear the PWM and a compare-up action to set the PWM. Modulate the compare value from 0 to TBPRD to achieve 50%-0% PWM duty.

When using up-count mode to generate an asymmetric PWM:

- To achieve 0-100% asymmetric PWM use the following configuration: Load CMPA/CMPB on TBPRD. Use the Zero action to set the PWM and a compare-up action to clear the PWM. Modulate the compare value from 0 to TBPRD+1 to achieve 0-100% PWM duty.

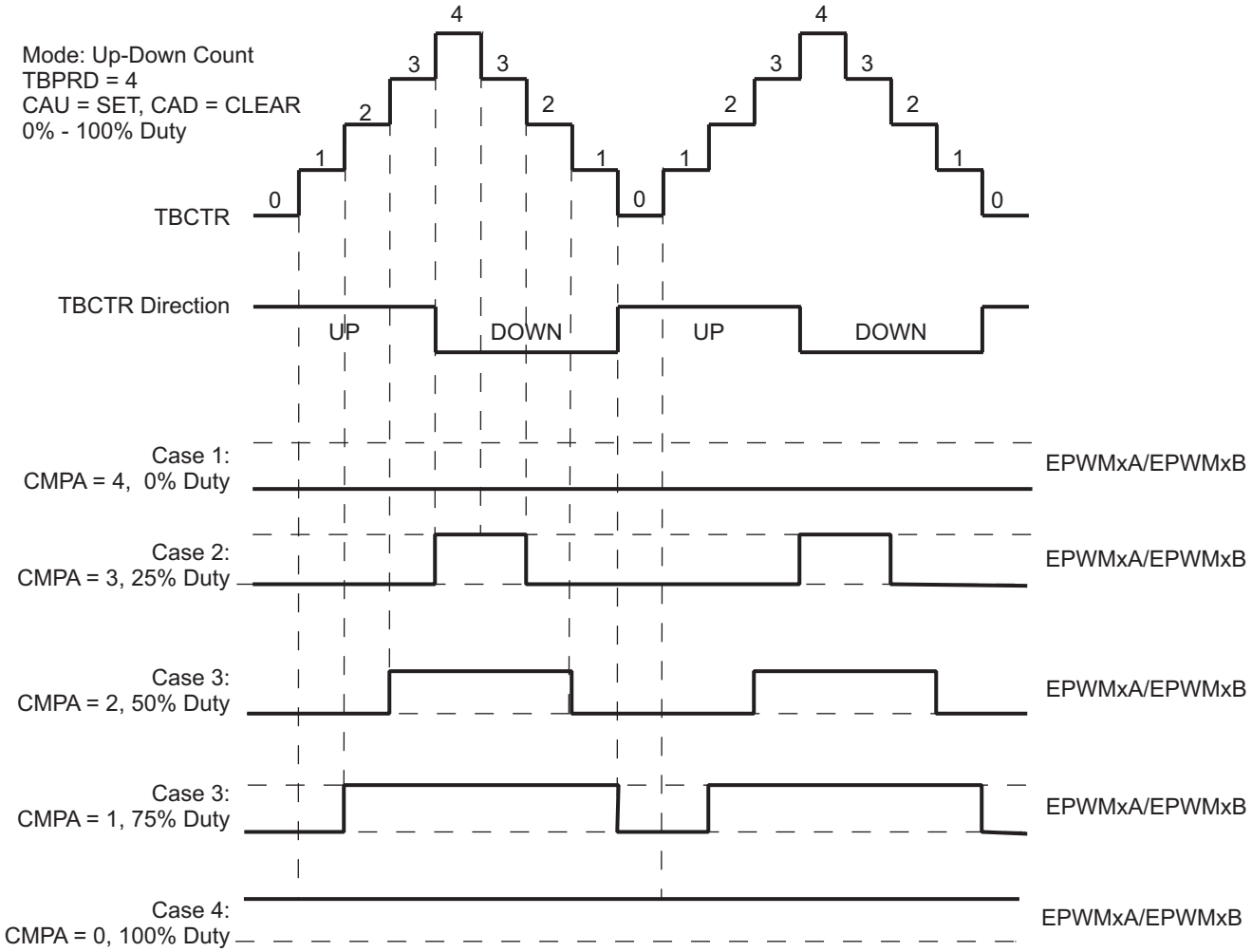
See the *Using Enhanced Pulse Width Modulator (ePWM) Module for 0-100% Duty Cycle Control* Application Report (literature number SPRAAI1)

Figure 7-23 shows how a symmetric PWM waveform can be generated using the up-down-count mode of the TBCTR. In this mode 0%-100% DC modulation is achieved by using equal compare matches on the up count and down count portions of the waveform. In the example shown, CMPA is used to make the comparison. When the counter is incrementing the CMPA match will pull the PWM output high. Likewise, when the counter is decrementing the compare match will pull the PWM signal low. When CMPA = 0, the PWM signal is low for the entire period giving the 0% duty waveform. When CMPA = TBPRD, the PWM signal is high achieving 100% duty.

When using this configuration in practice, if you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1. If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1. This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-23. Up-Down-Count Mode Symmetrical Waveform

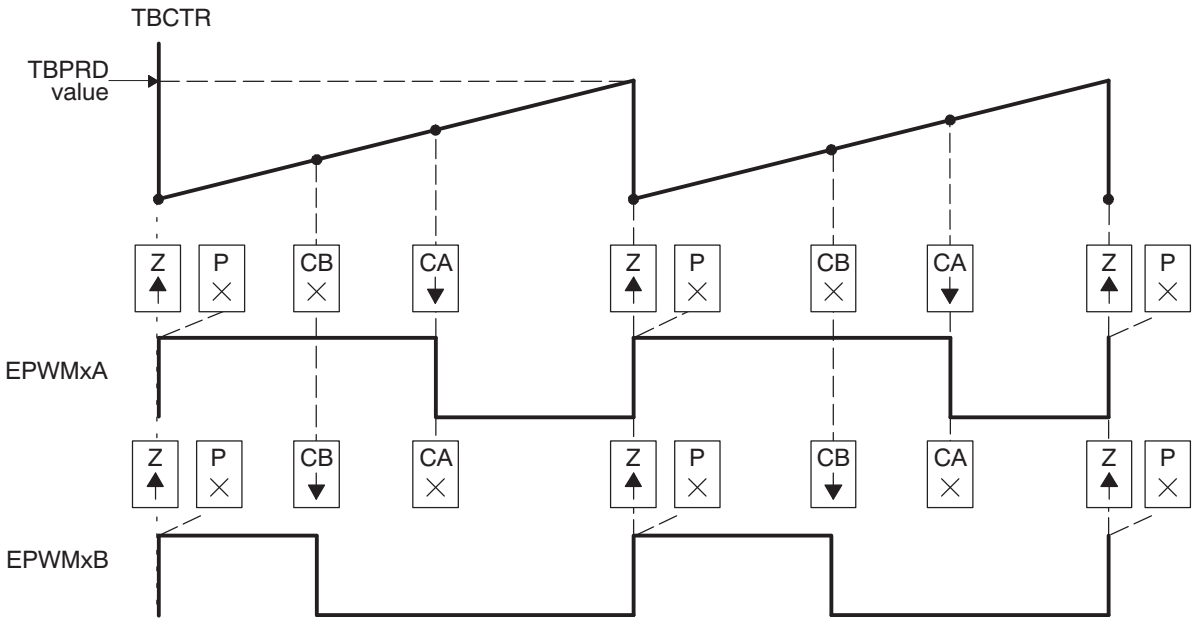


The PWM waveforms in Figure 7-24 through Figure 7-29 show some common action-qualifier configurations. The C-code samples in Example 7-1 through Example 7-6 shows how to configure an ePWM module for each case. Some conventions used in the figures and examples are as follows:

- TBPRD, CMPA, and CMPB refer to the value written in their respective registers. The active register, not the shadow register, is used by the hardware.
- CMPx, refers to either CMPA or CMPB.
- EPWMxA and EPWMxB refer to the output signals from ePWMx
- Up-Down means Count-up-and-down mode, Up means up-count mode and Dwn means down-count mode
- Sym = Symmetric, Asym = Asymmetric

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-24. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High



A PWM period = (TBPRD + 1) × T_{TBCLK}

B Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).

C Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).

D The "Do Nothing" actions (X) are shown for completeness, but will not be shown on subsequent diagrams.

E Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

Example 7-1 contains a code sample showing initialization and run time for the waveforms in Figure 7-24.

Example 7-1. Code Sample for Figure 7-24

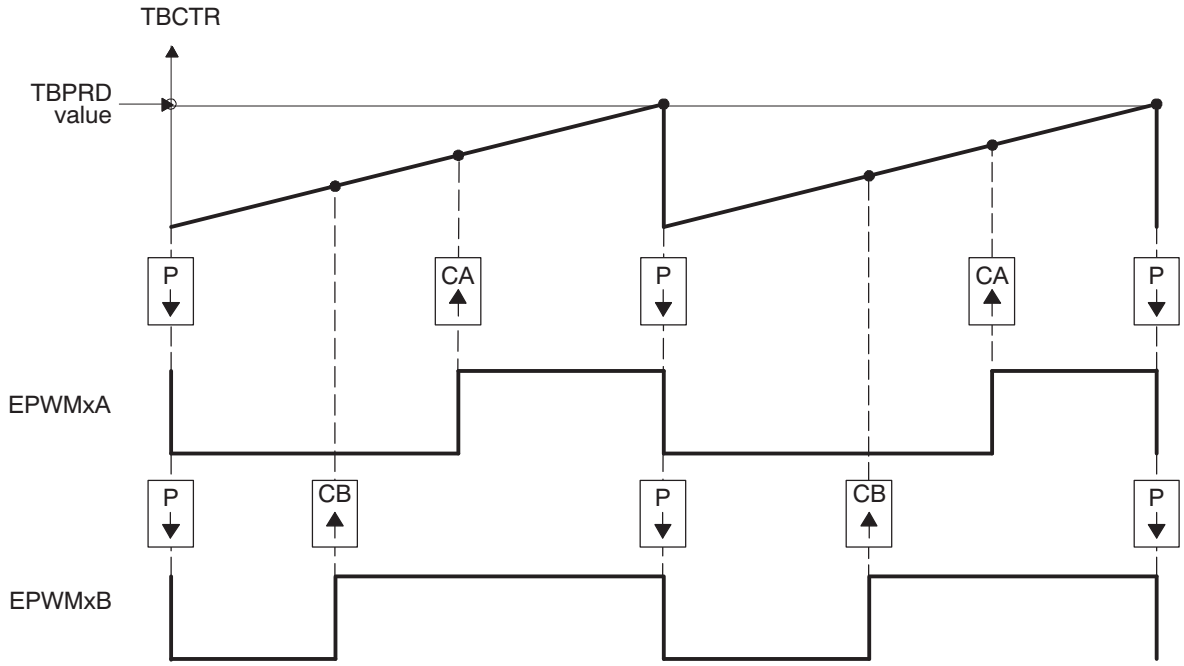
```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 350;                 // Compare A = 350 TBCLK counts
EPwm1Regs.CMPB = 200;                           // Compare B = 200 TBCLK counts
EPwm1Regs.TBPHS = 0;                            // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                            // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;        // TBCLK = SYSCLK
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = Duty1A;              // adjust duty for output EPWM1A
```


Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-1. Code Sample for Figure 7-24 (continued)

```
EPwm1Regs.CMPB = Duty1B;                    // adjust duty for output EPWM1B
```

Figure 7-25. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low



A PWM period = (TBPRD + 1) × T_{TBCLK}

B Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).

C Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).

D Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

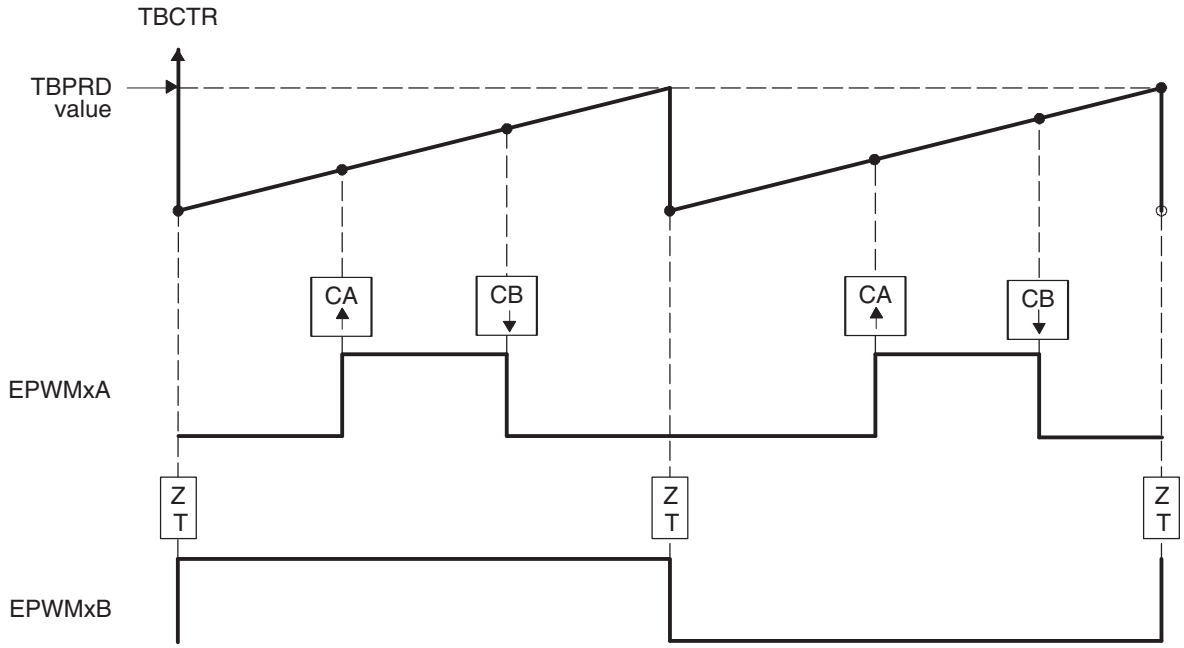
Example 7-2 contains a code sample showing initialization and run time for the waveforms in Figure 7-25.

Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-2. Code Sample for Figure 7-25

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 350;                 // Compare A = 350 TBCLK counts
EPwm1Regs.CMPB = 200;                           // Compare B = 200 TBCLK counts
EPwm1Regs.TBPHS = 0;                            // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                            // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;        // TBCLK = SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on TBCTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on TBCTR = Zero
EPwm1Regs.AQCTLA.bit.PRD = AQ_CLEAR;
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLB.bit.PRD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = Duty1A;              // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;                        // adjust duty for output EPWM1B
```

Figure 7-26. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA



A PWM frequency = 1/((TBPRD + 1) × T_{TBCLK})

B Pulse can be placed anywhere within the PWM cycle (0000 - TBPRD)

C High time duty proportional to (CMPB - CMPA)

D EPWMxB can be used to generate a 50% duty square wave with frequency = ◆ × ((TBPRD + 1) × TBCLK)

Example 7-3 contains a code sample showing initialization and run time for the waveforms Figure 7-26. Use the code in Example 7-5 to define the headers.

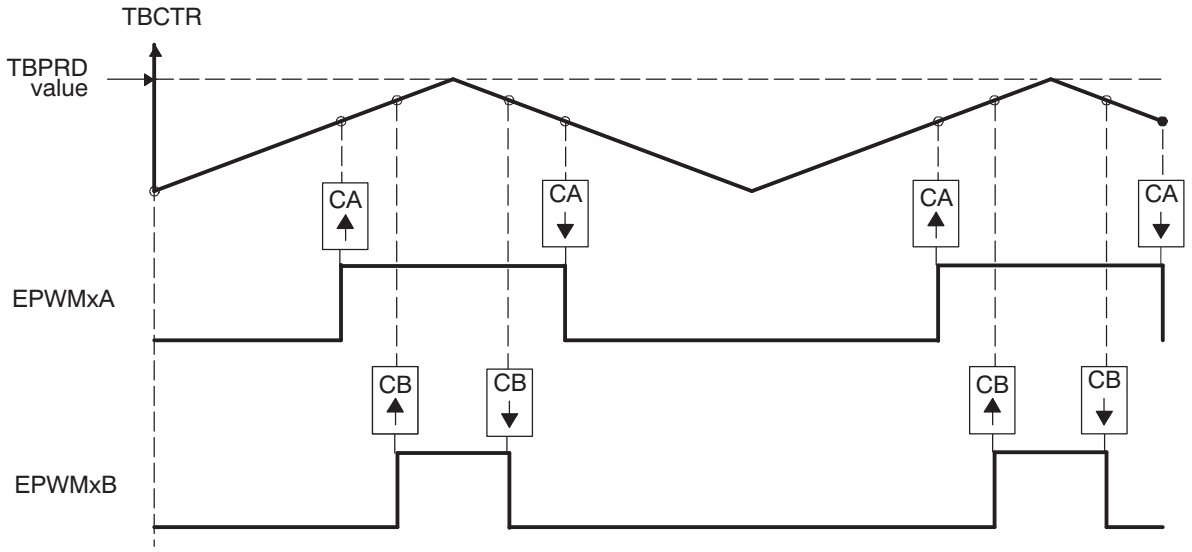
Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-3. Code Sample for Figure 7-26

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                        // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 200;               // Compare A = 200 TBCLK counts
EPwm1Regs.CMPB = 400;                         // Compare B = 400 TBCLK counts
EPwm1Regs.TBPHS = 0;                          // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                          // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;       // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;      // TBCLK = SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on TBCTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on TBCTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CBU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_TOGGLE;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = EdgePosA;         // adjust duty for output EPWM1A only
EPwm1Regs.CMPB = EdgePosB;
```


Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-27. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low



A PWM period = 2 x TBPRD × T_{TBCLK}

B Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).

C Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).

D Outputs EPWMxA and EPWMxB can drive independent power switches

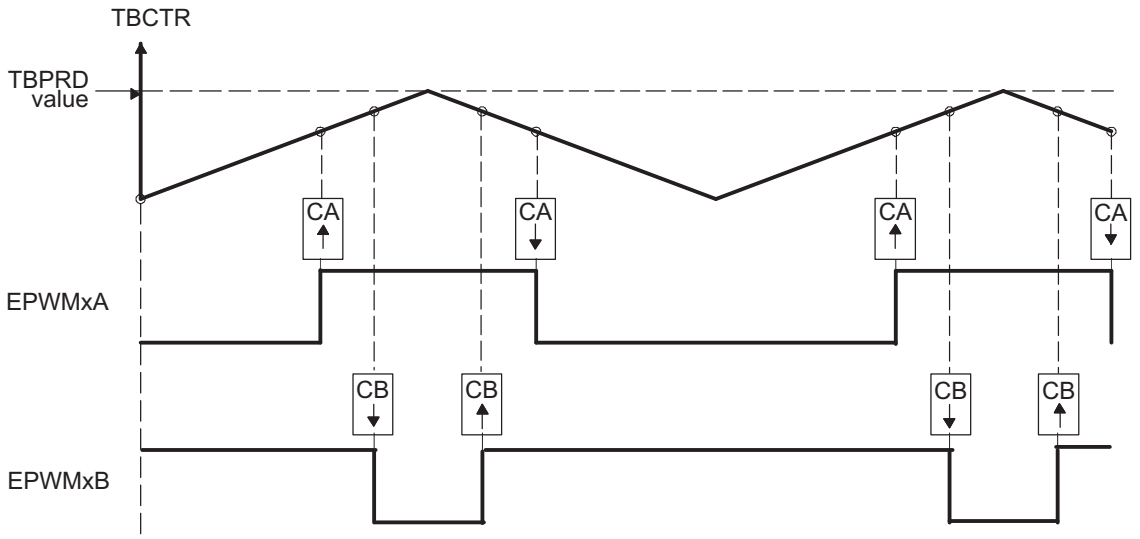
Example 7-4 contains a code sample showing initialization and run time for the waveforms in Figure 7-27. Use the code in Example 7-5 to define the headers.

Example 7-4. Code Sample for Figure 7-27

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          // Period = 2´600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 400;                 // Compare A = 400 TBCLK counts
EPwm1Regs.CMPB = 500;                           // Compare B = 500 TBCLK counts
EPwm1Regs.TBPHS = 0;                            // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                            // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetric
xEPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;        // Phase loading disabled
xEPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;        // TBCLK = SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;
EPwm1Regs.AQCTLB.bit.CBD = AQ_CLEAR;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = Duty1A;             // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;                       // adjust duty for output EPWM1B
```


Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-28. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary



A PWM period = 2 × TBPRD × T_{TBCLK}

B Duty modulation for EPWMxA is set by CMPA, and is active low, i.e., low time duty proportional to CMPA

C Duty modulation for EPWMxB is set by CMPB and is active high, i.e., high time duty proportional to CMPB

D Outputs EPWMx can drive upper/lower (complementary) power switches

E Dead-band = CMPB - CMPA (fully programmable edge placement by software). Note the dead-band module is also available if the more classical edge delay method is required.

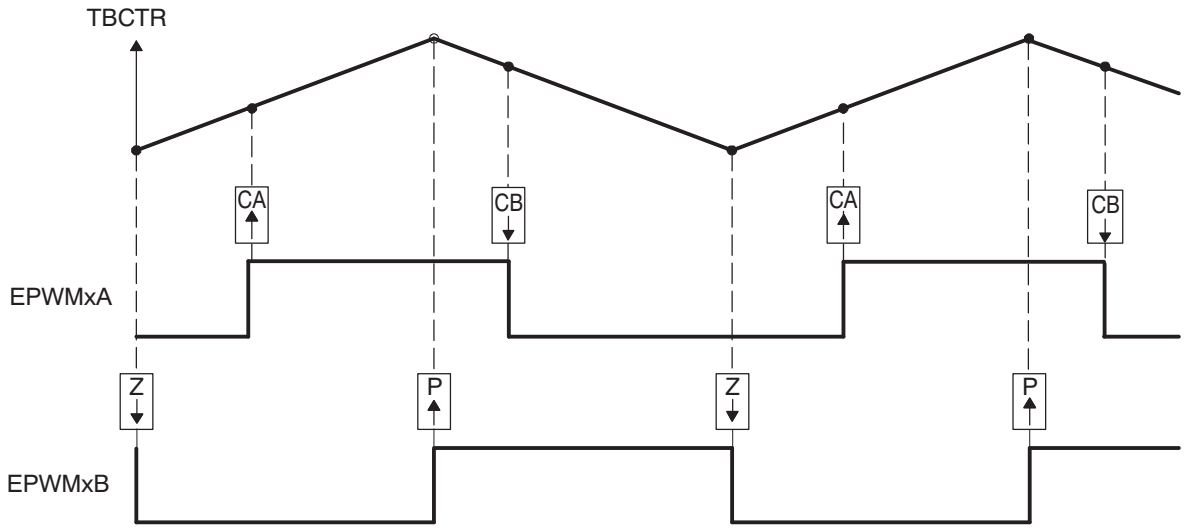
Example 7-5 contains a code sample showing initialization and run time for the waveforms in Figure 7-28. Use the code in Example 7-5 to define the headers.

Example 7-5. Code Sample for Figure 7-28

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          // Period = 2´600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 350;                 // Compare A = 350 TBCLK counts
EPwm1Regs.CMPB = 400;                           // Compare B = 400 TBCLK counts
EPwm1Regs.TBPHS = 0;                            // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                            // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetric
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;       // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;      // TBCLK = SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBD = AQ_SET;
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = Duty1A;          // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;                    // adjust duty for output EPWM1B
```


Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-29. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low



A PWM period = 2 × TBPRD × TBCLK

B Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.

C Duty modulation for EPWMxA is set by CMPA and CMPB.

D Low time duty for EPWMxA is proportional to (CMPA + CMPB).

E To change this example to active high, CMPA and CMPB actions need to be inverted (i.e., Set ! Clear and Clear Set).

F Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB)

Example 7-6 contains a code sample showing initialization and run time for the waveforms in Figure 7-29. Use the code in Example 7-5 to define the headers.

Example 7-6. Code Sample for Figure 7-29

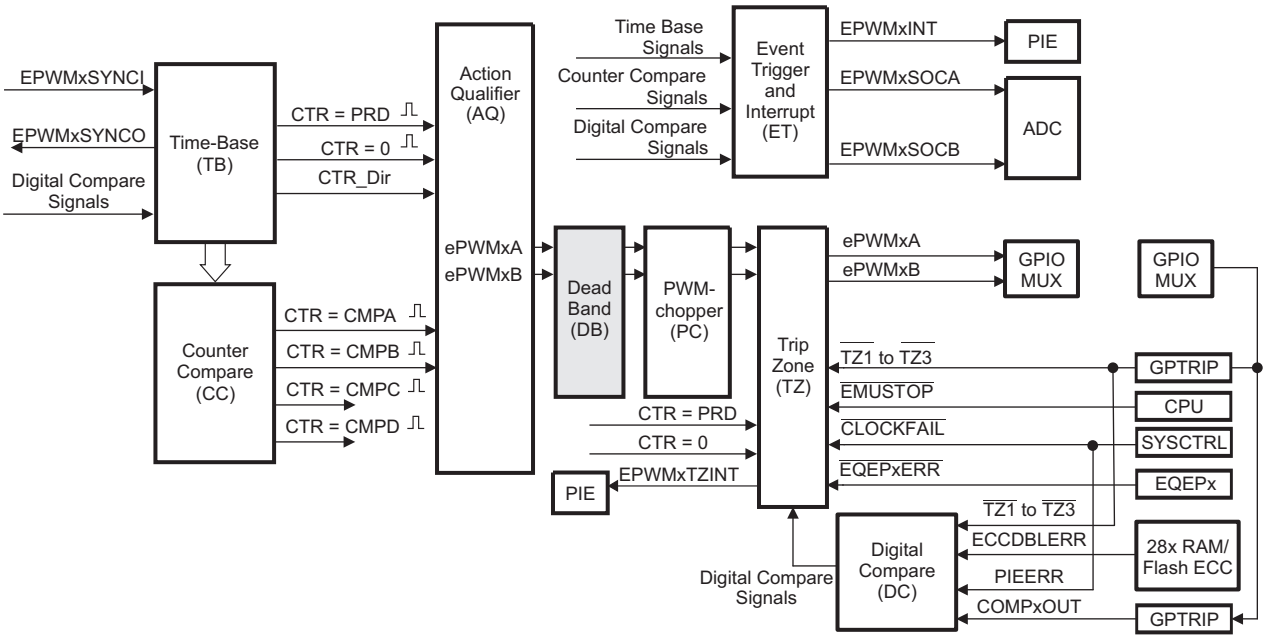
```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          //  Period = 2 ´ 600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 250;                 //  Compare A = 250 TBCLK counts
EPwm1Regs.CMPB = 450;                           //  Compare B = 450 TBCLK counts
EPwm1Regs.TBPHS = 0;                            //  Set Phase register to zero
EPwm1Regs.TBCTR = 0;                            //  clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; //  Symmetric
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         //  Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;        //  TBCLK = SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  //  load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  //  load on CTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CBD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.PRD = AQ_SET;
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = EdgePosA;            // adjust duty for output EPWM1A only
EPwm1Regs.CMPB = EdgePosB;
```


Copyright © 2012–2019, Texas Instruments Incorporated

7.2.5 Dead-Band Generator (DB) Submodule

Figure 7-30 illustrates the dead-band submodule within the ePWM module.

Figure 7-30. Dead_Band Submodule



7.2.5.1 Purpose of the Dead-Band Submodule

The "Action-qualifier (AQ) Module" section discussed how it is possible to generate the required dead-band by having full control over edge placement using both the CMPA and CMPB resources of the ePWM module. However, if the more classical edge delay-based dead-band with polarity control is required, then the dead-band submodule described here should be used.

The key functions of the dead-band module are:

- Generating appropriate signal pairs (EPWMxA and EPWMxB) with dead-band relationship from a single EPWMxA input
- Programming signal pairs for:
  - Active high (AH)
  - Active low (AL)
  - Active high complementary (AHC)
  - Active low complementary (ALC)
- Adding programmable delay to rising edges (RED)
- Adding programmable delay to falling edges (FED)
- Can be totally bypassed from the signal path (note dotted lines in diagram)

7.2.5.2 Dead band Submodule Additonal Operating Modes

On type 1 ePWM RED could appear on one channel output and FED could appear on the other channel output.

The following list shows the distinct difference between type 1 and type 2 modules with respect to Dead Band operating modes

Copyright © 2012–2019, Texas Instruments Incorporated

1. By adding S6, S7, and S8 in the figure "Configuration Options for the Dead-Band Submodule" below, RED and FED can appear on both the A-channel and B-channel outputs. Additionally, both RED and FED together can be applied to either the A-channel or B-channel outputs to allow B-channel phase shifting with respect to the A-channel. Note: Phase shifting B-channel with respect to the A-channel using the Dead band Submodule Additonal Operating Modes has limitations with respect to the choice of RED and FED delay with respect to the operating duty cycle of the ePWMxA and ePWMxB outputs.

2. The dead-band counters have also been increased to 14-bits

3. Dead-band and dead-band High-resolution registers are now shadowed.

4. High-resolution deadband RED and FED have been enabled using the DBREDHR and DBFEDHR registers

NOTE: The PWM-Chopper will not be enabled when high-resolution dead-band is enabled

NOTE: High-resolution deadband RED and FED requires Half-Cycle clocking mode (DBCTL[HALFCYCLE] = 1).

Cannot have both RED and FED together applied to both ePWMxA and ePWMxB. RED and FED together can be applied only to either OutA OR OutB.

NOTE: Phase shifting B-channel with respect to the A-channel: When PWMxB is derived from PWMxA using the DEDB_MODE bit and by delaying rising edge and falling edge by the phase shift amount. When the duty cycle value on PWMxA is less than this phase shift amount, PWMxA's falling edge has precedence over the delayed rising edge for PWMxB. It is recommended to make sure the duty cycle value of the current waveform fed to the Dead band module is greater than the required phase shift amount.

Shadow Mode:

The shadow mode for the DBRED is enabled by setting the DBCTL[SHDWDBREDMODE] bit and the shadow register for DBFED is enabled by setting the DBCTL[SHDWDBFEDMODE] bit. Shadow mode is disabled by default for both DBRED and DBFED

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events as specified by the DBCTL[LOADREDMODE] & DBCTL [LOADFEDMODE] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero

NOTE: When DBRED/DBFED active is loaded with a new shadow value while DB counters are counting, the new DBRED / DBFED value only affects the NEXT PWMx edge and not the current edge.

NOTE: If a shadow-to-active load for the DBRED or DBFED registers occurs at the same time as a corresponding rising or falling edge, then the RED or FED is applied to the PWM signal before the shadow-to-active load occurs. This causes shadow-to-active loads for the Dead-band's RED or FED to not take effect for this event. The value contained within the active register will take effect for this event; subsequent RED or FED delays will reflect this shadow-to-active load. Immediate writes to the RED or FED registers are unaffected

7.2.5.3 Controlling and Monitoring the Dead-Band Submodule

The dead-band submodule operation is controlled and monitored via the following registers:

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-14. Dead-Band Generator Submodule Registers

Register Name	Address Offset	Shadowed	Description
DBCTL	0x0F	No	Dead-Band Control Register
DBRED	0x10	No	Dead-Band Rising Edge Delay Count Register
DBFED	0x11	No	Dead-Band Falling Edge Delay Count Register
DBREDHR	0x6C	Yes	Dead-Band Generator Rising Edge Delay High Resolution Mirror Register
DBREDM	0x6D	Yes	Dead-Band Generator Rising Edge Delay Count Mirror Register
DBFEDHR	0x6E	Yes	Dead-Band Generator Falling Edge Delay High Resolution Register
DBFEDM	0x6F	Yes	Dead-Band Generator Falling Edge Delay Count Register

7.2.5.4 Operational Highlights for the Dead-Band Submodule

The following sections provide the operational highlights.

The dead-band submodule has two groups of independent selection options as shown in Figure 7-31.

- **Input Source Selection:**

  The input signals to the dead-band module are the EPWMxA and EPWMxB output signals from the action-qualifier. In this section they will be referred to as EPWMxA In and EPWMxB In. Using the DBCTL[IN_MODE) control bits, the signal source for each delay, falling-edge or rising-edge, can be selected:

  – EPWMxA In is the source for both falling-edge and rising-edge delay. This is the default mode.
  – EPWMxA In is the source for falling-edge delay, EPWMxB In is the source for rising-edge delay.
  – EPWMxA In is the source for rising edge delay, EPWMxB In is the source for falling-edge delay.
  – EPWMxB In is the source for both falling-edge and rising-edge delay.

- **Half Cycle Clocking:**

  The dead-band submodule can be clocked using half cycle clocking to double the resolution (i.e. counter clocked at 2× TBCLK)

- **Output Mode Control:**

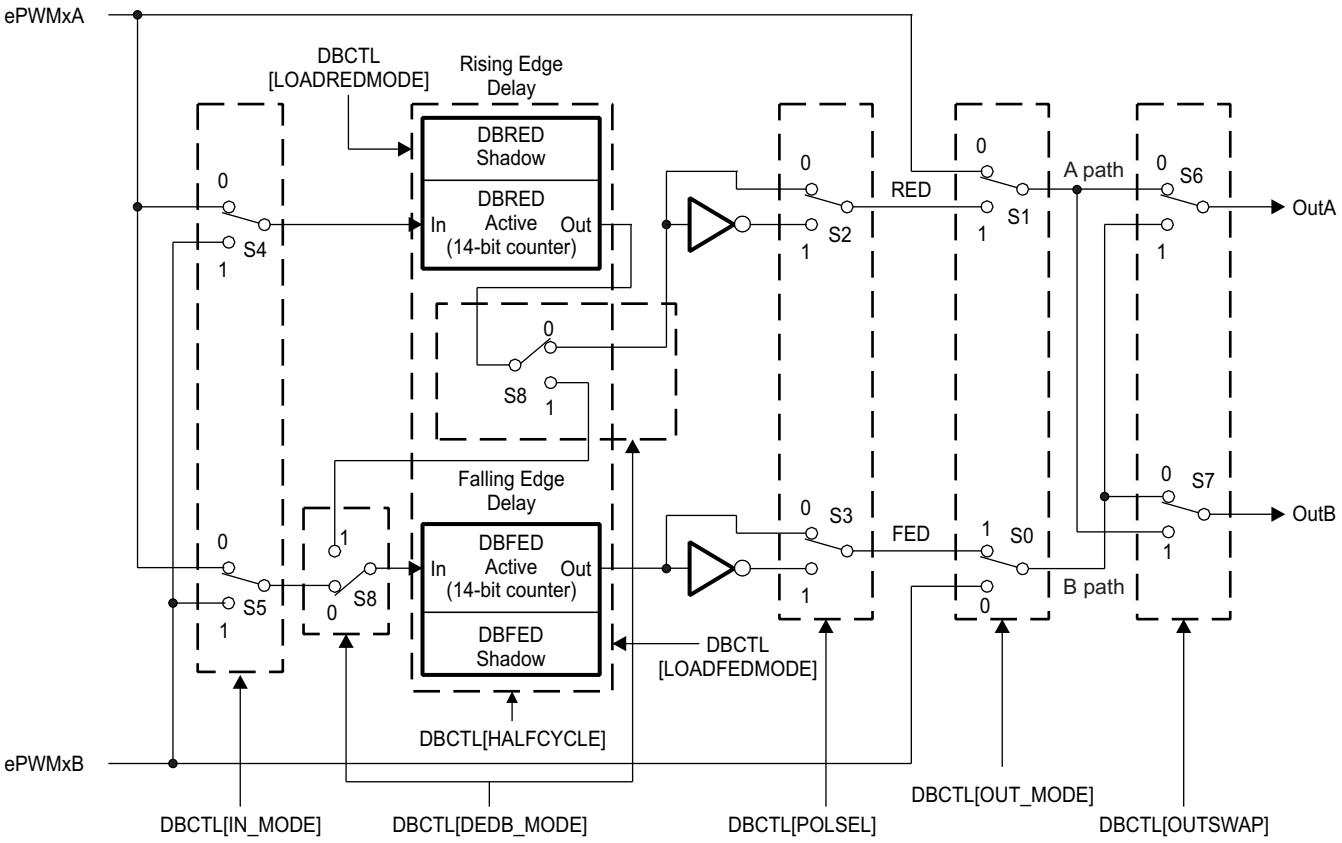
  The output mode is configured by way of the DBCTL[OUT_MODE] bits. These bits determine if the falling-edge delay, rising-edge delay, neither, or both are applied to the input signals.

- **Polarity Control:**

  The polarity control (DBCTL[POLSEL]) allows you to specify whether the rising-edge delayed signal and/or the falling-edge delayed signal is to be inverted before being sent out of the dead-band submodule.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-31. Configuration Options for the Dead-Band Submodule



Although all combinations are supported, not all are typical usage modes. Table 7-15 documents some classical dead-band configurations. These modes assume that the DBCTL[IN_MODE] is configured such that EPWMxA In is the source for both falling-edge and rising-edge delay. Enhanced, or non-traditional modes can be achieved by changing the input signal source. The modes shown in Table 7-15 fall into the following categories:

- **Mode 1: Bypass both falling-edge delay (FED) and rising-edge delay (RED)**

  Allows you to fully disable the dead-band submodule from the PWM signal path.
- **Mode 2-5: Classical Dead-Band Polarity Settings:**

  These represent typical polarity configurations that should address all the active high/low modes required by available industry power switch gate drivers. The waveforms for these typical cases are shown in Figure 7-32. Note that to generate equivalent waveforms to Figure 7-32, configure the action-qualifier submodule to generate the signal as shown for EPWMxA.
- **Mode 6: Bypass rising-edge-delay and Mode 7: Bypass falling-edge-delay**

  Finally the last two entries in Table 7-15 show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

Table 7-15. Classical Dead-Band Operating Modes

Mode	Mode Description	DBCTL[POLSEL]		DBCTL[OUT_MODE]	
		S3	S2	S1	S0
1	EPWMxA and EPWMxB Passed Through (No Delay)	X	X	0	0
2	Active High Complementary (AHC)	1	0	1	1
3	Active Low Complementary (ALC)	0	1	1	1
4	Active High (AH)	0	0	1	1
5	Active Low (AL)	1	1	1	1

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-15. Classical Dead-Band Operating Modes (continued)

Mode	Mode Description	DBCTL[POLSEL]		DBCTL[OUT_MODE]	
		S3	S2	S1	S0
6	EPWMxA Out = EPWMxA In (No Delay) EPWMxB Out = EPWMxA In with Falling Edge Delay	0 or 1	0 or 1	0	1
7	EPWMxA Out = EPWMxA In with Rising Edge Delay EPWMxB Out = EPWMxB In with No Delay	0 or 1	0 or 1	1	0

Table 7-16. Additional Dead-Band Operating Modes

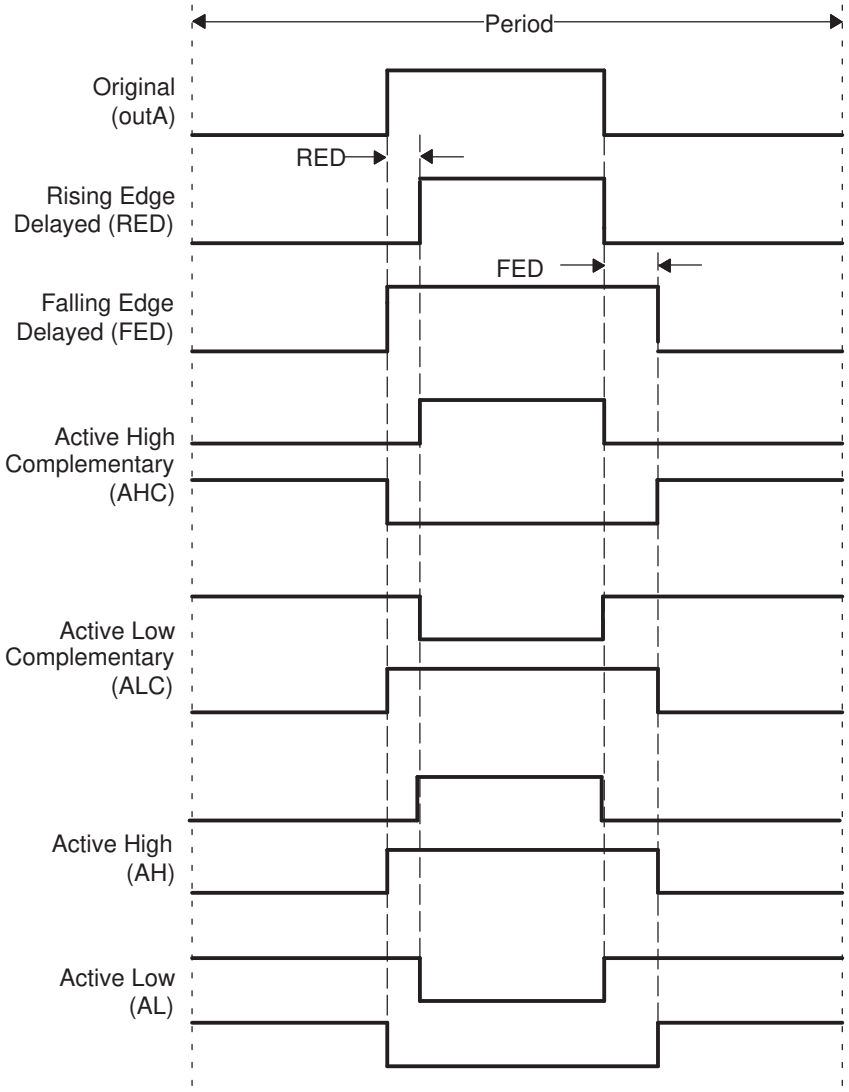
Mode Description	DBCTL[DEDB-MODE]	DBCTL[OUTSWAP]	
	S8	S6	S7
EPWMxA and EPWMxB signals are as defined by OUT-MODE bits.	0	0	0
EPWMxA = A-path as defined by OUT-MODE bits. EPWMxB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A-signal path)	0	0	1
EPWMxA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B-signal path) EPWMxB = B-path as defined by OUT-MODE bits	0	1	0
EPWMxA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B-signal path) EPWMxB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A-signal path)	0	1	1
Rising edge delay applied to EPWMxA / EPWMxB as selected by S4 switch (IN-MODE bits) on A signal path only. Falling edge delay applied to EPWMxA / EPWMxB as selected by S5 switch (IN-MODE bits) on B signal path only.	0	X	X
Rising edge delay and falling edge delay applied to source selected by S4 switch (IN-MODE bits) and output to B signal path only.[(1)]	1	X	X

(1) When this bit is set to 1, user should always either set OUT_MODE bits such that Apath = InA ***or*** OUTSWAP bits such that EPWMxA=Bpath. Otherwise, EPWMxA will be invalid.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-32 shows waveforms for typical cases where 0% < duty < 100%.

Figure 7-32. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)



Copyright © 2012–2019, Texas Instruments Incorporated

The dead-band submodule supports independent values for rising-edge (RED) and falling-edge (FED) delays. The amount of delay is programmed using the DBRED and DBFED registers. These are 10-bit registers and their value represents the number of time-base clock, TBCLK, periods a signal edge is delayed by. For example, the formula to calculate falling-edge-delay and rising-edge-delay are:

$FED = DBFED \times T_{TBCLK}$

$RED = DBRED \times T_{TBCLK}$

Where T_{TBCLK} is the period of TBCLK, the prescaled version of SYSCLKOUT.

For convenience, delay values for various TBCLK options are shown in Table 7-17.

Table 7-17. Dead-Band Delay Values in μS as a Function of DBFED and DBRED

Dead-Band Value	Dead-Band Delay in μS		
DBFED, DBRED	TBCLK = SYSCLKOUT/1	TBCLK = SYSCLKOUT /2	TBCLK = SYSCLKOUT/4
1	0.01 μS	0.03 μS	0.05 μS
5	0.06 μS	0.13μS	0.25 μS
10	0.13 μS	0.25 μS	0.50 μS
100	1.25 μS	2.50 μS	5.00 μS
200	2.50 μS	5.00 μS	10.00 μS
400	5.00 μS	10.00 μS	20.00 μS
500	6.25 μS	12.50 μS	25.00 μS
600	7.50 μS	15.00 μS	30.00 μS
700	8.75 μS	17.50 μS	35.00 μS
800	10.00 μS	20.00 μS	40.00 μS
900	11.25 μS	22.50 μS	45.00 μS
1000	12.50 μS	25.00 μS	50.00 μS

When half-cycle clocking is enabled, the formula to calculate the falling-edge-delay and rising-edge-delay becomes:

$FED = DBFED \times T_{TBCLK}/2$

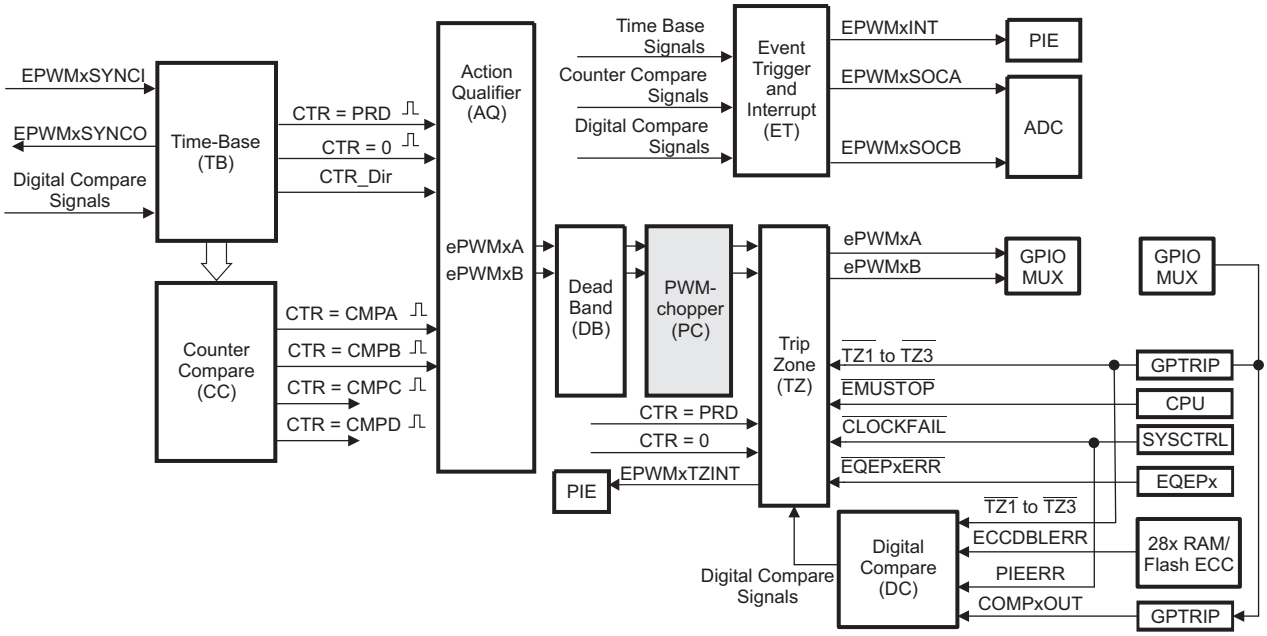
$RED = DBRED \times T_{TBCLK}/2$

Copyright © 2012–2019, Texas Instruments Incorporated

7.2.6 PWM-Chopper (PC) Submodule

Figure 7-33 illustrates the PWM-chopper (PC) submodule within the ePWM module.

Figure 7-33. PWM-Chopper Submodule



The PWM-chopper submodule allows a high-frequency carrier signal to modulate the PWM waveform generated by the action-qualifier and dead-band submodules. This capability is important if you need pulse transformer-based gate drivers to control the power switching elements.

7.2.6.1 Purpose of the PWM-Chopper Submodule

The key functions of the PWM-chopper submodule are:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

7.2.6.2 Controlling the PWM-Chopper Submodule

The PWM-chopper submodule operation is controlled via the registers in Table 7-18.

Table 7-18. PWM-Chopper Submodule Registers

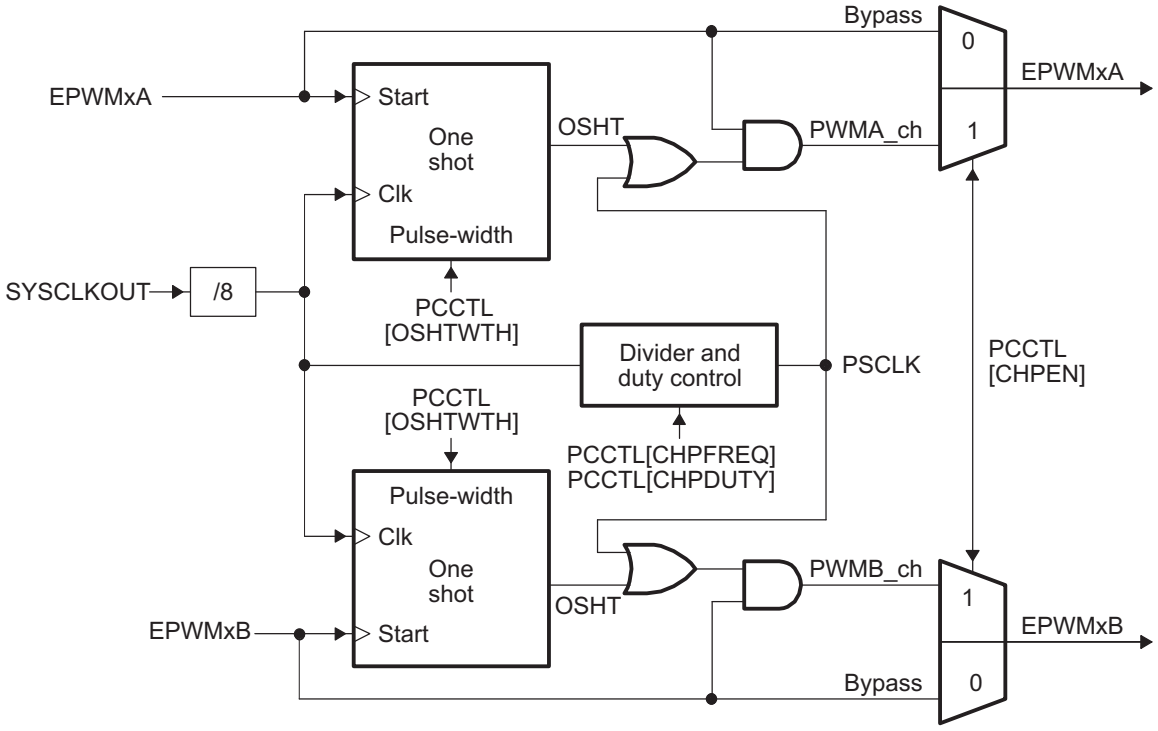
Register Name	Address Offset	Shadowed	Description
PCCTL	0x1E	No	PWM-Chopper Control Register

7.2.6.3 Operational Highlights for the PWM-Chopper Submodule

Figure 7-34 shows the operational details of the PWM-chopper submodule. The carrier clock is derived from SYSCLKOUT. Its frequency and duty cycle are controlled via the CHPFREQ and CHPDUTY bits in the PCCTL register. The one-shot block is a feature that provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on. The one-shot width is programmed via the OSHTWTH bits. The PWM-chopper submodule can be fully disabled (bypassed) via the CHPEN bit.

Copyright © 2012–2019, Texas Instruments Incorporated

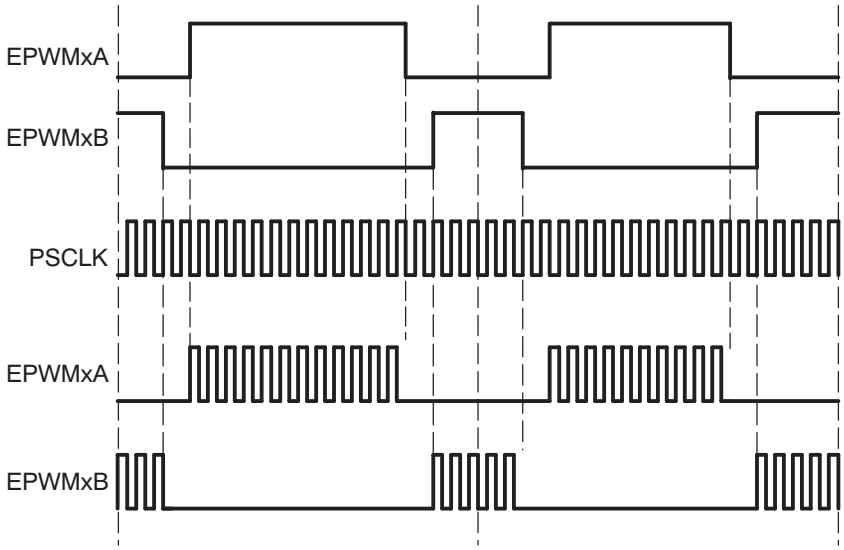
Figure 7-34. PWM-Chopper Submodule Operational Details



7.2.6.4 Waveforms

Figure 7-35 shows simplified waveforms of the chopping action only; one-shot and duty-cycle control are not shown. Details of the one-shot and duty-cycle control are discussed in the following sections.

Figure 7-35. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only



Copyright © 2012–2019, Texas Instruments Incorporated

7.2.6.4.1 One-Shot Pulse

The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$T_{1stpulse} = T_{SYSCLKOUT} \times 8 \times OSHTWTH$

Where $T_{SYSCLKOUT}$ is the period of the system clock (SYSCLKOUT) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 7-36 shows the first and subsequent sustaining pulses and Table 7.3 gives the possible pulse width values for a SYSCLKOUT = 80 MHz.

Figure 7-36. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses

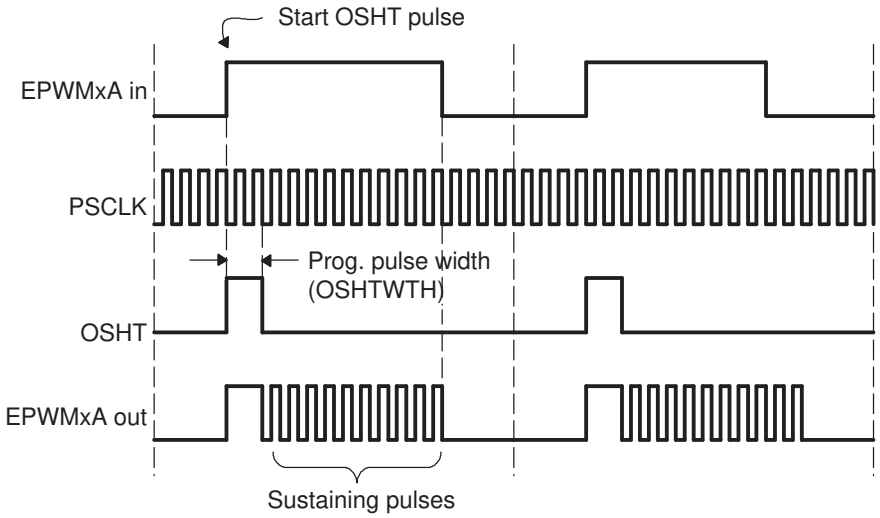


Table 7-19. Possible Pulse Width Values for SYSCLKOUT = 80 MHz

OSHTWTHz (hex)	Pulse Width (nS)
0	100
1	200
2	300
3	400
4	500
5	600
6	700
7	800
8	900
9	1000
A	1100
B	1200
C	1300
D	1400
E	1500
F	1600

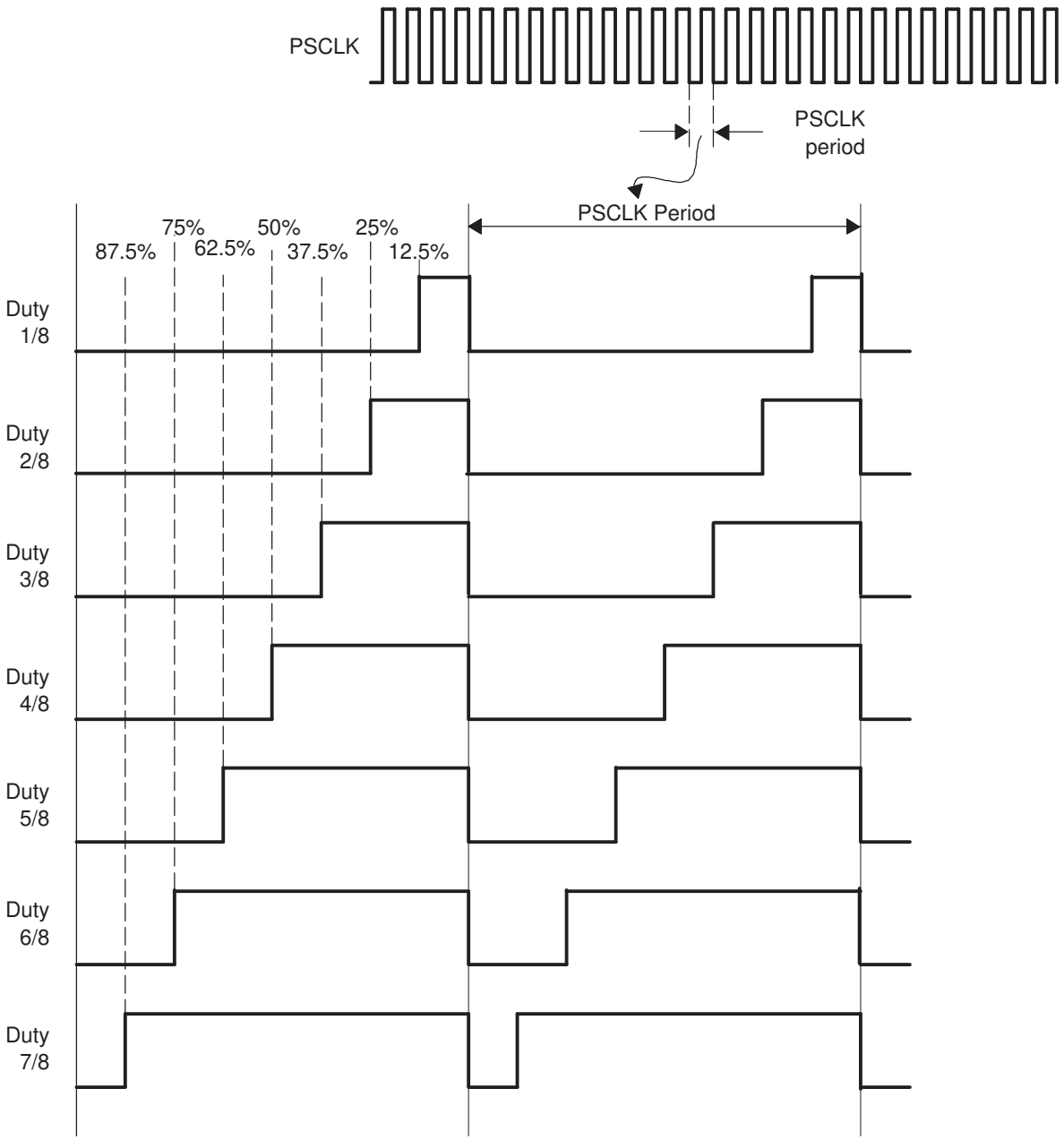
Copyright © 2012–2019, Texas Instruments Incorporated

7.2.6.4.2 Duty Cycle Control

Pulse transformer-based gate drive designs need to comprehend the magnetic properties or characteristics of the transformer and associated circuitry. Saturation is one such consideration. To assist the gate drive designer, the duty cycles of the second and subsequent pulses have been made programmable. These sustaining pulses ensure the correct drive strength and polarity is maintained on the power switch gate during the on period, and hence a programmable duty cycle allows a design to be tuned or optimized via software control.

Figure 7-37 shows the duty cycle control that is possible by programming the CHPDUTY bits. One of seven possible duty ratios can be selected ranging from 12.5% to 87.5%.

Figure 7-37. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses

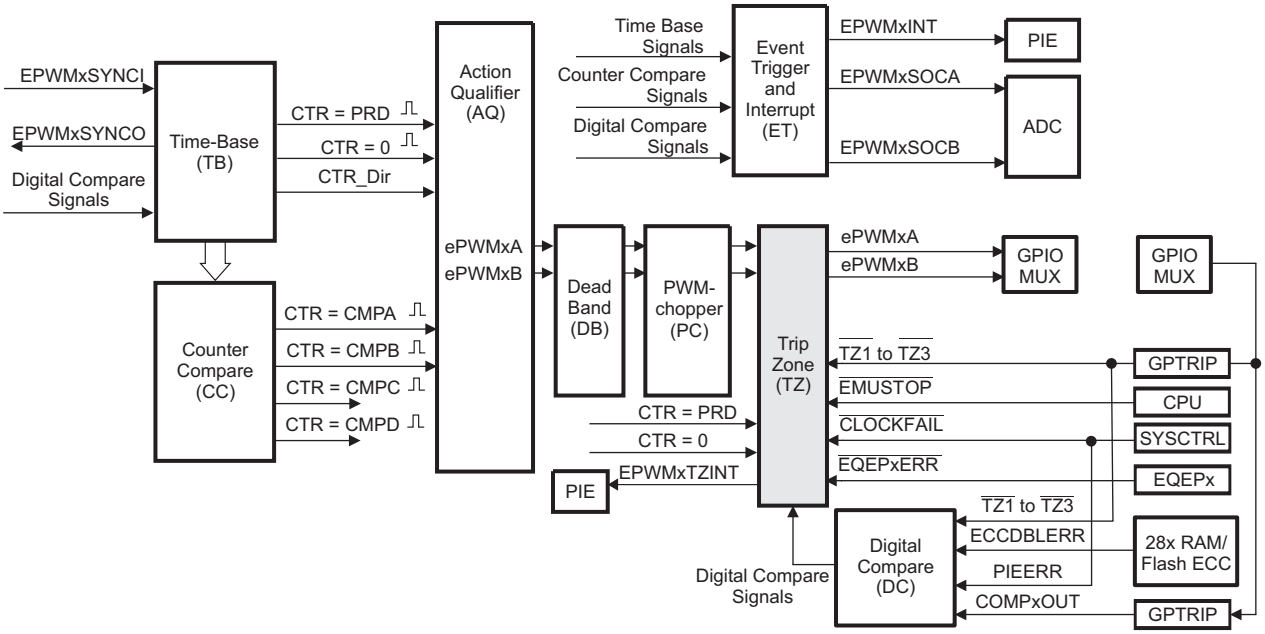


Copyright © 2012–2019, Texas Instruments Incorporated

7.2.7 Trip-Zone (TZ) Submodule

Figure 7-38 shows how the trip-zone (TZ) submodule fits within the ePWM module.

Figure 7-38. Trip-Zone Submodule



Each ePWM module is connected to six $\overline{\text{TZn}}$ signals ($\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$). $\overline{\text{TZ1}}$ to $\overline{\text{TZ3}}$ are sourced from the GPIO mux. $\overline{\text{TZ4}}$ is sourced from an inverted EQEPxERR signal on those devices with an EQEP module. $\overline{\text{TZ5}}$ is connected to the system clock fail logic, and $\overline{\text{TZ6}}$ is sourced from the EMUSTOP output from the CPU. These signals indicate external fault or trip conditions, and the ePWM outputs can be programmed to respond accordingly when faults occur.

7.2.7.1 Purpose of the Trip-Zone Submodule

The key functions of the Trip-Zone submodule are:

- Trip inputs $\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$ can be flexibly mapped to any ePWM module.
- Upon a fault condition, outputs EPWMxA and EPWMxB can be forced to one of the following:
  - High
  - Low
  - High-impedance
  - No action taken
- Support for one-shot trip (OSHT) for major short circuits or over-current conditions.
- Support for cycle-by-cycle tripping (CBC) for current limiting operation.
- Support for digital compare tripping (DC) based on state of on-chip analog comparator module outputs and/or $\overline{\text{TZ1}}$ to $\overline{\text{TZ3}}$ signals.
- Each trip-zone input and digital compare (DC) submodule DCAEVT1/2 or DCBEVT1/2 force event can be allocated to either one-shot or cycle-by-cycle operation.
- Interrupt generation is possible on any trip-zone input.
- Software-forced tripping is also supported.
- The trip-zone submodule can be fully bypassed if it is not required.

Copyright © 2012–2019, Texas Instruments Incorporated

7.2.7.2 Controlling and Monitoring the Trip-Zone Submodule

The trip-zone submodule operation is controlled and monitored through the following registers:

Table 7-20. Trip-Zone Submodule Registers

Register Name	Address Offset	Shadowed	Description [1]
TZSEL	0x12	No	Trip-Zone Select Register
TZDCSEL	0x13	No	Trip-Zone Digital Compare Select Register [2]
TZCTL	0x14	No	Trip-Zone Control Register
TZEINT	0x15	No	Trip-Zone Enable Interrupt Register
TZFLG	0x16	No	Trip-Zone Flag Register
TZCLR	0x17	No	Trip-Zone Clear Register
TZFRC	0x18	No	Trip-Zone Force Register
TZCLRM	0x71	No	Trip-Zone Clear Register Mirror

(1) All trip-zone registers are EALLOW protected and can be modified only after executing the EALLOW instruction. For more information, see the *System Control and Interrupts Reference Guide*.

(2) This register is discussed in more detail in Section 7.2.9, *Digital Compare (DC) Submodule*.

7.2.7.3 Operational Highlights for the Trip-Zone Submodule

The following sections describe the operational highlights and configuration options for the trip-zone submodule.

The trip-zone signals $\overline{TZ1}$ to $\overline{TZ6}$ (also collectively referred to as $\overline{TZn}$) are active low input signals. When one of these signals goes low, or when a DCAEVT1/2 or DCBEVT1/2 force happens based on the TZDCSEL register event selection, it indicates that a trip event has occurred. Each ePWM module can be individually configured to ignore or use each of the trip-zone signals or DC events. Which trip-zone signals or DC events are used by a particular ePWM module is determined by the TZSEL register for that specific ePWM module. The trip-zone signals may or may not be synchronized to the system clock (SYSCLKOUT) and digitally filtered within the GPIO MUX block. A minimum of 3*TBCLK low pulse width on $\overline{TZn}$ inputs is sufficient to trigger a fault condition on the ePWM module. If the pulse width is less than this, the trip condition may not be latched by CBC or OST latches. The asynchronous trip makes sure that if clocks are missing for any reason, the outputs can still be tripped by a valid event present on $\overline{TZn}$ inputs . The GPIOs or peripherals must be appropriately configured. For more information, see the device-specific version of the *System Control and Interrupts Reference Guide*.

Each $\overline{TZn}$ input can be individually configured to provide either a cycle-by-cycle or one-shot trip event for an ePWM module. DCAEVT1 and DCBEVT1 events can be configured to directly trip an ePWM module or provide a one-shot trip event to the module. Likewise, DCAEVT2 and DCBEVT2 events can also be configured to directly trip an ePWM module or provide a cycle-by-cycle trip event to the module. This configuration is determined by the TZSEL[DCAEVT1/2], TZSEL[DCBEVT1/2], TZSEL[CBCn], and TZSEL[OSHTn] control bits (where n corresponds to the trip input) respectively.

- **Cycle-by-Cycle (CBC):**

  When a cycle-by-cycle trip event occurs, the action specified in the TZCTL[TZA] and TZCTL[TZB] bits is carried out immediately on the EPWMxA and/or EPWMxB output. Table 7-21 lists the possible actions. In addition, the cycle-by-cycle trip event flag (TZFLG[CBC]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral.

  If the CBC interrupt is enabled via the TZEINT register, and DCAEVT2 or DCBEVT2 are selected as CBC trip sources via the TZSEL register, it is not necessary to also enable the DCAEVT2 or DCBEVT2 interrupts in the TZEINT register, as the DC events trigger interrupts through the CBC mechanism.

  The specified condition on the inputs is automatically cleared based on the selection made with TZCLR[CBCPULSE] if the trip event is no longer present. Therefore, in this mode, the trip event is cleared or reset every PWM cycle. The TZFLG[CBC] flag bit will remain set until it is manually cleared by writing to the TZCLR[CBC] bit. If the cycle-by-cycle trip event is still present when the TZFLG[CBC] bit is cleared, then it will again be immediately set.

- **One-Shot (OSHT):**

  When a one-shot trip event occurs, the action specified in the TZCTL[TZA] and TZCTL[TZB] bits is

Copyright © 2012–2019, Texas Instruments Incorporated

carried out immediately on the EPWMxA and/or EPWMxB output. Table 7-21 lists the possible actions. In addition, the one-shot trip event flag (TZFLG[OST]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral. The one-shot trip condition must be cleared manually by writing to the TZCLR[OST] bit.

If the one-shot interrupt is enabled via the TZEINT register, and DCAEVT1 or DCBEVT1 are selected as OSHT trip sources via the TZSEL register, it is not necessary to also enable the DCAEVT1 or DCBEVT1 interrupts in the TZEINT register, as the DC events trigger interrupts through the OSHT mechanism.

- **Digital Compare Events (DCAEVT1/2 and DCBEVT1/2):**

  A digital compare DCAEVT1/2 or DCBEVT1/2 event is generated based on a combination of the DCAH/DCAL and DCBH/DCBL signals as selected by the TZDCSEL register. The signals which source the DCAH/DCAL and DCBH/DCBL signals are selected via the DCTRIPSEL register and can be either trip zone input pins or analog comparator COMPxOUT signals. For more information on the digital compare submodule signals, see Section 7.2.9.

  When a digital compare event occurs, the action specified in the TZCTL[DCAEVT1/2] and TZCTL[DCBEVT1/2] bits is carried out immediately on the EPWMxA and/or EPWMxB output. Table 7-21 lists the possible actions. In addition, the relevant DC trip event flag (TZFLG[DCAEVT1/2] / TZFLG[DCBEVT1/2]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral.

  The specified condition on the pins is automatically cleared when the DC trip event is no longer present. The TZFLG[DCAEVT1/2] or TZFLG[DCBEVT1/2] flag bit will remain set until it is manually cleared by writing to the TZCLR[DCAEVT1/2] or TZCLR[DCBEVT1/2] bit. If the DC trip event is still present when the TZFLG[DCAEVT1/2] or TZFLG[DCBEVT1/2] flag is cleared, then it will again be immediately set.

The action taken when a trip event occurs can be configured individually for each of the ePWM output pins by way of the TZCTL register bit fields. One of four possible actions, shown in Table 7-21, can be taken on a trip event.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-21. Possible Actions On a Trip Event

TZCTL Register bit-field Settings	EPWMxA and/or EPWMxB	Comment
0,0	High-Impedance	Tripped
0,1	Force to High State	Tripped
1,0	Force to Low State	Tripped
1,1	No Change	Do Nothing. No change is made to the output.

Example 7-7. Trip-Zone Configurations

Scenario A:
A one-shot trip event on $\overline{TZ1}$ pulls both EPWM1A, EPWM1B low and also forces EPWM2A and EPWM2B high.

- Configure the ePWM1 registers as follows:
  - TZSEL[OSHT1] = 1: enables $\overline{TZ1}$ as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables $\overline{TZ1}$ as a one-shot event source for ePWM2
  - TZCTL[TZA] = 1: EPWM2A will be forced high on a trip event.
  - TZCTL[TZB] = 1: EPWM2B will be forced high on a trip event.

Scenario B:
A cycle-by-cycle event on $\overline{TZ5}$ pulls both EPWM1A, EPWM1B low.
A one-shot event on $\overline{TZ1}$ or $\overline{TZ6}$ puts EPWM2A into a high impedance state.

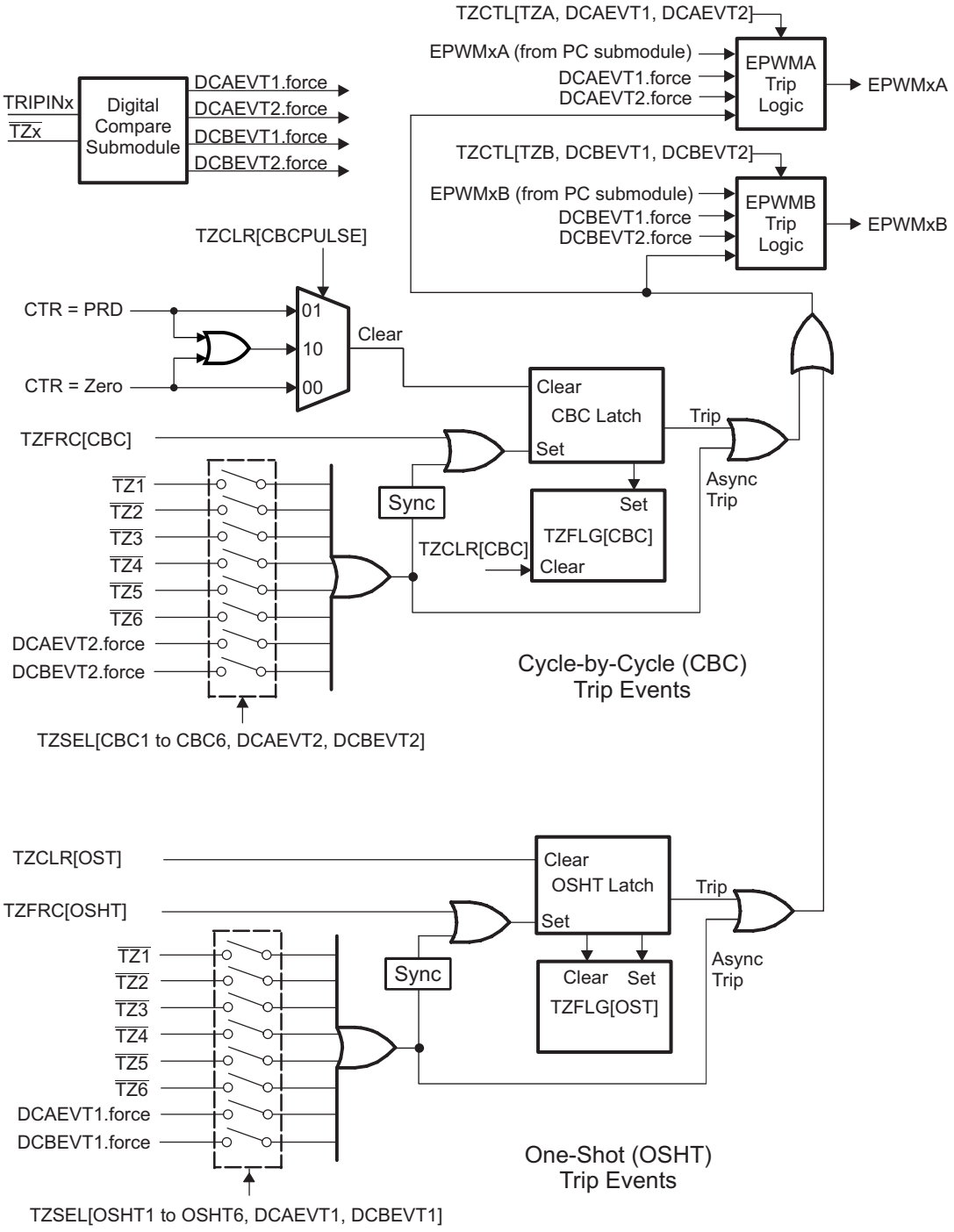
- Configure the ePWM1 registers as follows:
  - TZSEL[CBC5] = 1: enables $\overline{TZ5}$ as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables $\overline{TZ1}$ as a one-shot event source for ePWM2
  - TZSEL[OSHT6] = 1: enables $\overline{TZ6}$ as a one-shot event source for ePWM2
  - TZCTL[TZA] = 0: EPWM2A will be put into a high-impedance state on a trip event.
  - TZCTL[TZB] = 3: EPWM2B will ignore the trip event.

7.2.7.4 Generating Trip Event Interrupts

Figure 7-39 and Figure 7-40 illustrate the trip-zone submodule control and interrupt logic, respectively. DCAEVT1/2 and DCBEVT1/2 signals are described in further detail in Section 7.2.9.

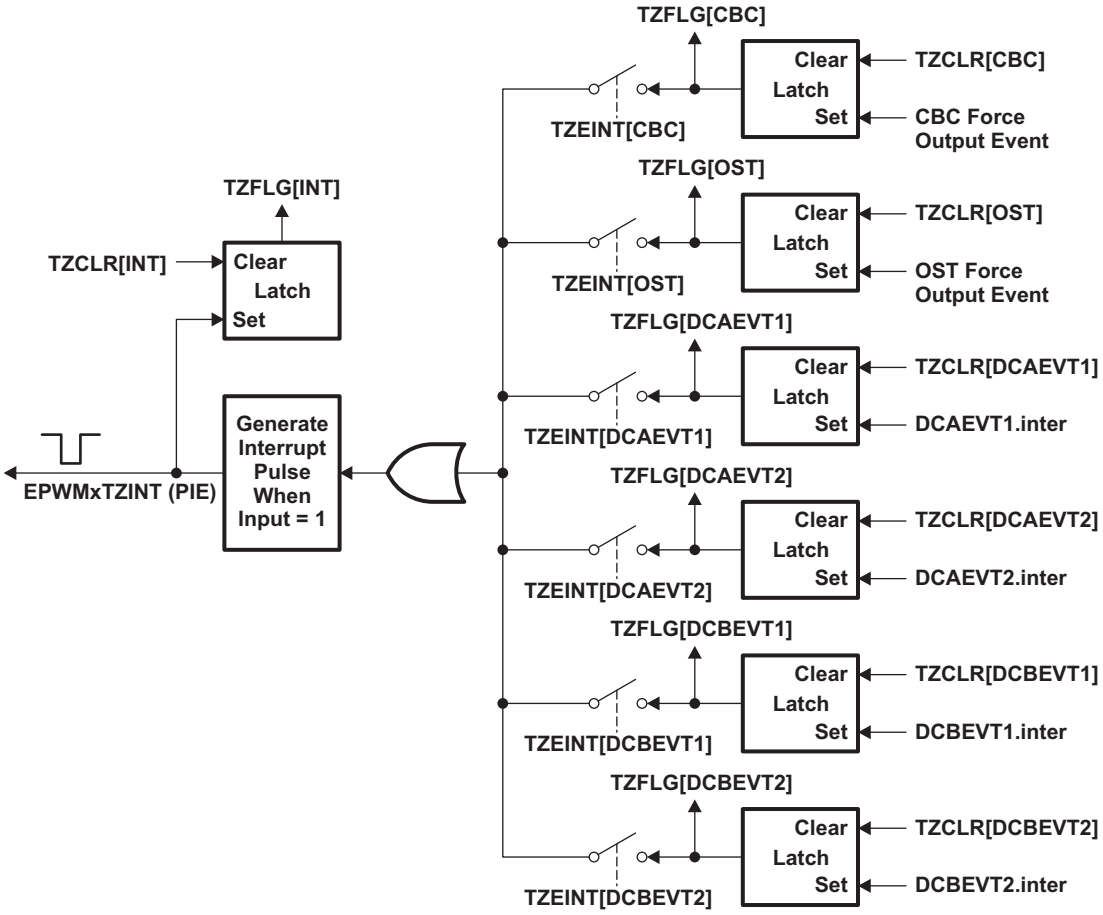
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-39. Trip-Zone Submodule Mode Control Logic



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-40. Trip-Zone Submodule Interrupt Logic



7.2.8 Event-Trigger (ET) Submodule

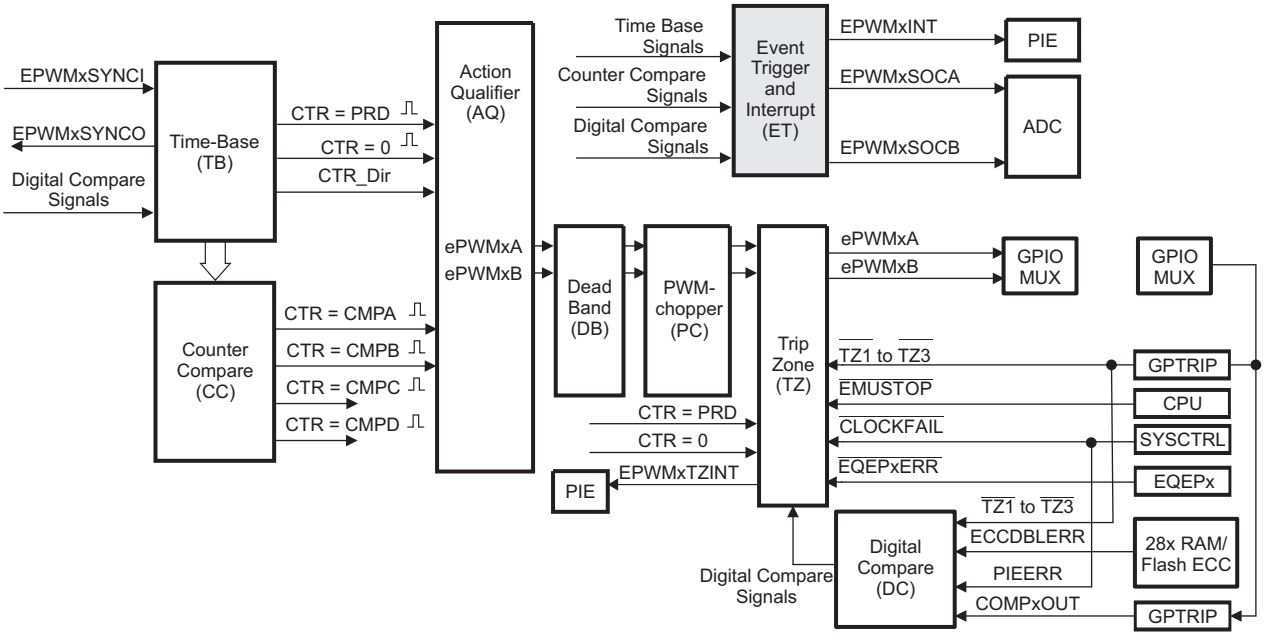
The key functions of the event-trigger submodule are:

- Receives event inputs generated by the time-base, counter-compare, and digital-compare submodules
- Uses the time-base direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests and ADC start of conversion at:
  - Every event
  - Every second event
  - Every third event
- Provides full visibility of event generation via event counters and flags
- Allows software forcing of Interrupts and ADC start of conversion

The event-trigger submodule manages the events generated by the time-base submodule, the counter-compare submodule, and the digital-compare submodule to generate an interrupt to the CPU and/or a start of conversion pulse to the ADC when a selected event occurs. Figure 7-41 illustrates where the event-trigger submodule fits within the ePWM system.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-41. Event-Trigger Submodule



7.2.8.1 Operational Overview of the ePWM type 2 Event-Trigger Submodule

The following sections describe the event-trigger submodule's operational highlights.

Each ePWM module has one interrupt request line connected to the PIE and two start of conversion signals connected to the ADC module. As shown in Figure 7-42, ADC start of conversion for all ePWM modules are connected to individual ADC trigger inputs to the ADC via the Analog Common Interface Bus (ACIB) and hence multiple modules can initiate an ADC start of conversion via the ADC trigger inputs. The configuration options for the TRIGxSEL register is shown in the below table.The trigger source for an ADC start of conversion is configured by a combination of the TRIGSEL field in the ADCSOCxCTL register, the appropriate bits in the ADCINTSOCSEL1 or ADCINTSOCSEL2 register, and setting the correct bits in the TRIGxSEL registers.

Table 7-22. TRIGxSEL Trigger Options

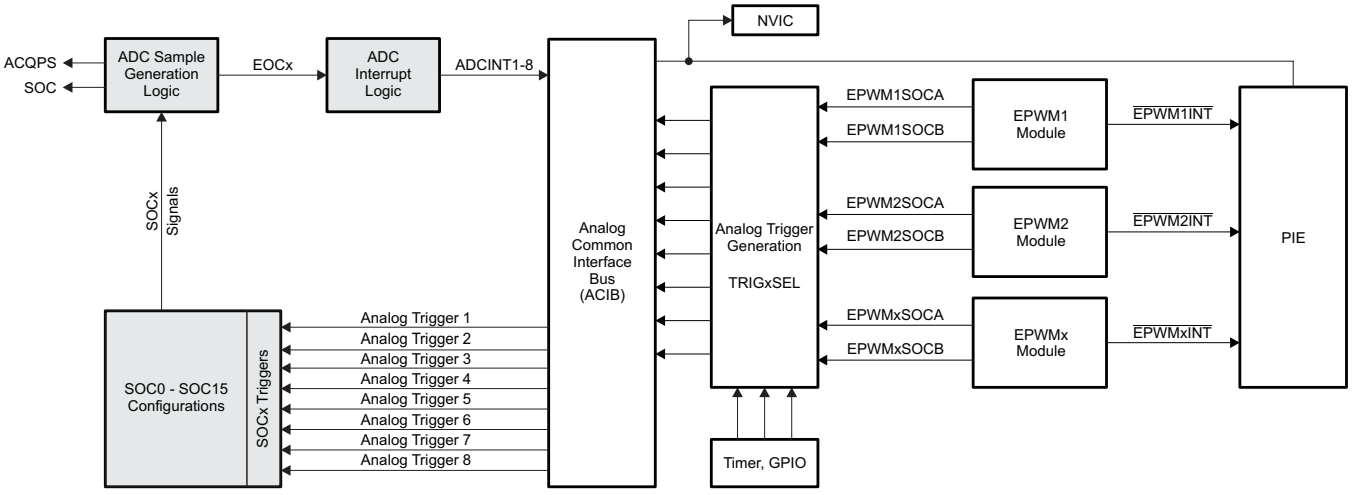
TRIGxSEL Bits	Trigger Source	Peripheral
00000	No Trigger Enabled (Default)	
00001	TINT0	CPU Timer 0
00010	TINT1	CPU Timer 1
00011	TINT2	CPU Timer 2
00100	XINT2	C28 GPIO MUX
00101	EPWM1SOCA	EPWM1
00110	EPWM1SOCB	
00111	EPWM1SYNC	
01000	EPWM2SOCA	EPWM2
01001	EPWM2SOCB	
01010	EPWM2SYNC	
01011	EPWM3SOCA	EPWM3
01100	EPWM3SOCB	
01101	EPWM3SYNC	

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-22. TRIGxSEL Trigger Options (continued)

TRIGxSEL Bits	Trigger Source	Peripheral
01110	EPWM4SOCA	EPWM4
01111	EPWM4SOCB	
10000	EPWM4SYNC	
10001	EPWM5SOCA	EPWM5
10010	EPWM5SOCB	
10011	EPWM5SYNC	
10100	EPWM6SOCA	EPWM6
10101	EPWM6SOCB	
10110	EPWM6SYNC	
10111	EPWM7SOCA	EPWM7
11000	EPWM7SOCB	
11001	EPWM7SYNC	
11010	EPWM8SOCA	EPWM8
11011	EPWM8SOCB	
11100	EPWM8SYNC	
11101	EPWM9SOCA	EPWM9
11110	EPWM9SOCB	
11111	EPWM9SYNC	

Figure 7-42. Event-Trigger Submodule Inter-Connectivity of ADC Start of Conversion

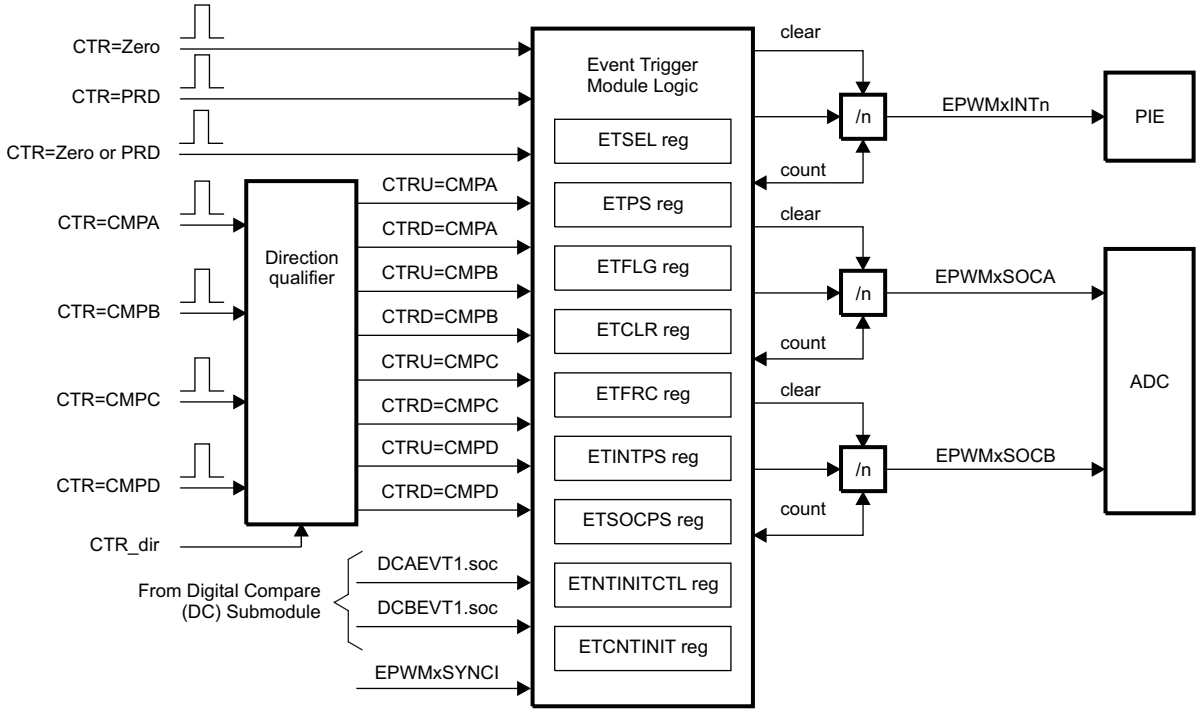


The event-trigger submodule monitors various event conditions (shown as inputs on the left side of Figure 7-43) and can be configured to prescale these events before issuing an Interrupt request or an ADC start of conversion. The event-trigger prescaling logic can issue Interrupt requests and ADC start of conversion at:

- Every event
- Every second event
- Up to Every fifteenth event

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-43. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs



The key registers used to configure the event-trigger submodule are shown in Table 7-23:

Table 7-23. Event-Trigger Submodule Registers

Register Name	Address Offset	Shadowed	Description
ETSEL	0x19	No	Event-Trigger Selection Register
ETPS	0x1A	No	Event-Trigger Prescale Register
ETFLG	0x1B	No	Event-Trigger Flag Register
ETCLR	0x1C	No	Event-Trigger Clear Register
ETFRC	0x1D	No	Event-Trigger Force Register
ETCLRM	0x70	No	Event-Trigger Clear Register Mirror
ETINTPS	0x50	No	Event-Trigger Interrupt Pre-Scale Register
ETSOCPS	0x51	No	Event-Trigger SOC Pre-Scale Register
ETCNTINITCTL	0x52	No	Event-Trigger Counter Initialization Control Register
ETCNTINIT	0x53	No	Event-Trigger Counter Initialization Register

- ETSEL - This selects which of the possible events will trigger an interrupt or start an ADC conversion.
- ETPS - This programs the event prescaling options mentioned above.
- ETFLG - These are flag bits indicating status of the selected and prescaled events.
- ETCLR - These bits allow you to clear the flag bits in the ETFLG register via software.
- ETFRC - These bits allow software forcing of an event. Useful for debugging or software intervention.
- ETINTPS - This programs the interrupt event prescaling options, supporting count and period up to 15 events.
- ETSOCPS - This programs the SOC event prescaling options, supporting count and period up to 15 events.
- ETCNTINITCTL - These bits enable ETCNTINIT initialization via SYNC event OR via software force.
- ETCNTINIT - These bits allow you to initialize INT/SOCA/SOCB counters on SYNC events (or software force) with user programmed value.

Copyright © 2012–2019, Texas Instruments Incorporated

A more detailed look at how the various register bits interact with the Interrupt and ADC start of conversion logic are shown in Figure 7-44, Figure 7-45, and Figure 7-46.

Figure 7-44 shows the event-trigger's interrupt generation logic. The interrupt-period (ETPS[INTPRD]) bits specify the number of events required to cause an interrupt pulse to be generated. The choices available are:

- Do not generate an interrupt.
- Generate an interrupt on every event
- Generate an interrupt on every second event
- Generate an interrupt on every third event

On ePWM type 2, in order to enable event generation capability up to 15 events the following changes have been made. The selection made on ETPS[INTPSSEL] bit determines whether ETINTPS register, INTCNT2 and INTPRD2 bit fields determine frequency of events (interrupt once every 0-15 events).

Which event can cause an interrupt is configured by the interrupt selection (ETSEL[INTSEL]) and (ETSEL[INTSELCMP]) bits. The event can be one of the following:

- Time-base counter equal to zero (TBCTR = 0x00).
- Time-base counter equal to period (TBCTR = TBPRD).
- Time-base counter equal to zero or period (TBCTR = 0x00 || TBCTR = TBPRD).
- Time-base counter equal to the compare A register (CMPA) when the timer is incrementing.
- Time-base counter equal to the compare A register (CMPA) when the timer is decrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is incrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is decrementing.
- Time-base counter equal to the compare C register (CMPC) when the timer is incrementing.
- Time-base counter equal to the compare C register (CMPC) when the timer is decrementing.
- Time-base counter equal to the compare D register (CMPD) when the timer is incrementing.
- Time-base counter equal to the compare D register (CMPD) when the timer is decrementing.

The number of events that have occurred can be read from the interrupt event counter ETPS[INTCNT] or ETINTPS[INTCNT2] register bits based off of the selection made using ETPS[INTPSSEL]. That is, when the specified event occurs the ETPS[INTCNT] or ETINTPS[INTCNT2] bits are incremented until they reach the value specified by ETPS[INTPRD] or ETINTPS[INTPRD2] determined again by the selection made in ETPS[INTPSSEL]. When ETPS[INTCNT] = ETPS[INTPRD] the counter stops counting and its output is set. The counter is only cleared when an interrupt is sent to the PIE.

When ETPS[INTCNT] reaches ETPS[INTPRD] the following behavior will occur [The below behavior is also applicable to ETINTPS[INTCNT2] & ETINTPS[INTPRD2] :

- If interrupts are enabled, ETSEL[INTEN] = 1 and the interrupt flag is clear, ETFLG[INT] = 0, then an interrupt pulse is generated and the interrupt flag is set, ETFLG[INT] = 1, and the event counter is cleared ETPS[INTCNT] = 0. The counter will begin counting events again.
- If interrupts are disabled, ETSEL[INTEN] = 0, or the interrupt flag is set, ETFLG[INT] = 1, the counter stops counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
- If interrupts are enabled, but the interrupt flag is already set, then the counter will hold its output high until the ENTFLG[INT] flag is cleared. This allows for one interrupt to be pending while one is serviced.

Writing to the INTPRD bits in different situations will cause the following results:

- Writing an INTPRD value that is GREATER or equal to the current counter value will reset the INTCNT = 0
- Writing an INTPRD value that is equal to the current counter value will trigger an interrupt if it is enabled and the status flag is cleared (and INTCNT will also be cleared to 0)
- Writing an INTPRD value that is LESS than the current counter value will result in undefined behavior (that is, INTCNT stops counting because INTPRD is below INTCNT, and interrupt will never fire).

Copyright © 2012–2019, Texas Instruments Incorporated

Writing a 1 to the ETFRC[INT] bit will increment the event counter INTCNT. The counter will behave as described above when INTCNT = INTPRD. When INTPRD = 0, the counter is disabled and hence no events will be detected and the ETFRC[INT] bit is also ignored. The same applies to ETINTPS[INTCNT2] & ETINTPS[INTPRD2]

The above definition means that you can generate an interrupt on every event, on every second event, or on every third event if using the INTCNT and INTPRD. You can generate an interrupt on every event up to 15 events if using the INTCNT2 and INTPRD2.

The INTCNT2 value can be initialized with the value from ETCNTINIT[INTINIT] based on the selection made in ETCNTINITCTL[INTINITEN]. When ETCNTINITCTL[INTINITEN] is set, then it enables initialization of INTCNT2 counter with contents of ETCNTINIT[INTINIT] on a SYNC event or software force determined by ETCNTINITCTL[INTINITFRC] .

Figure 7-44. Event-Trigger Interrupt Generator

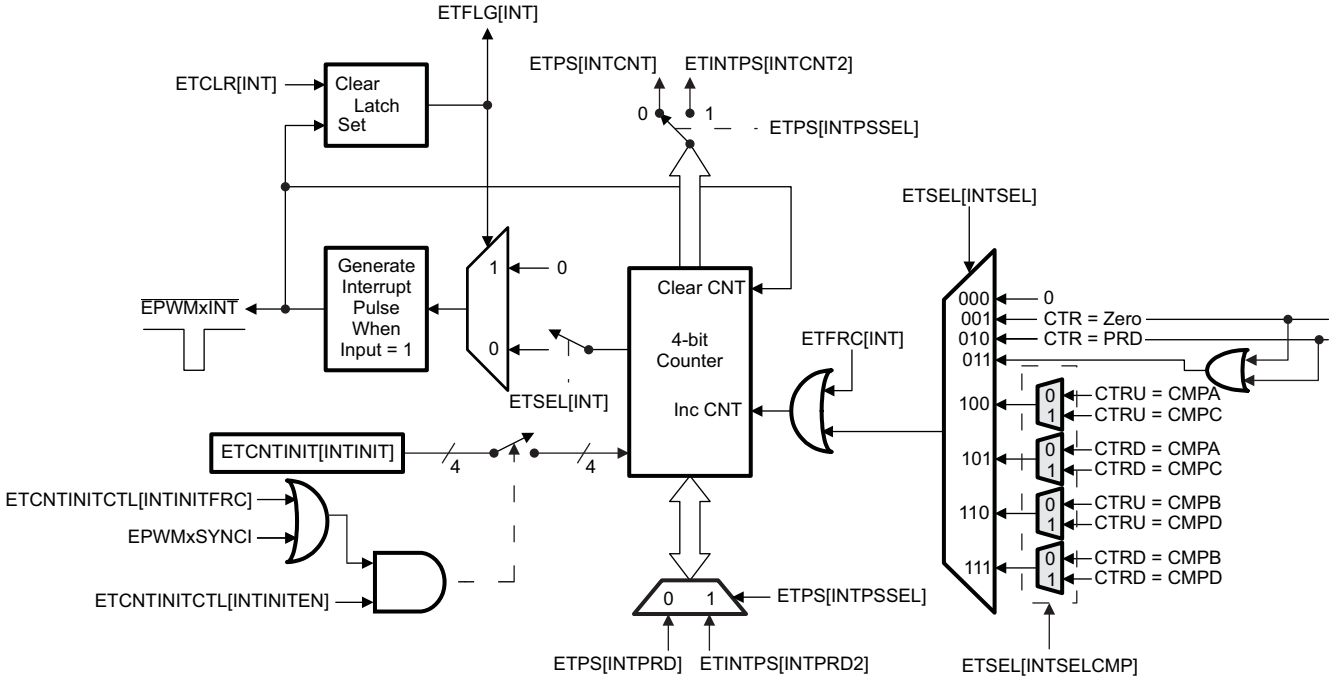
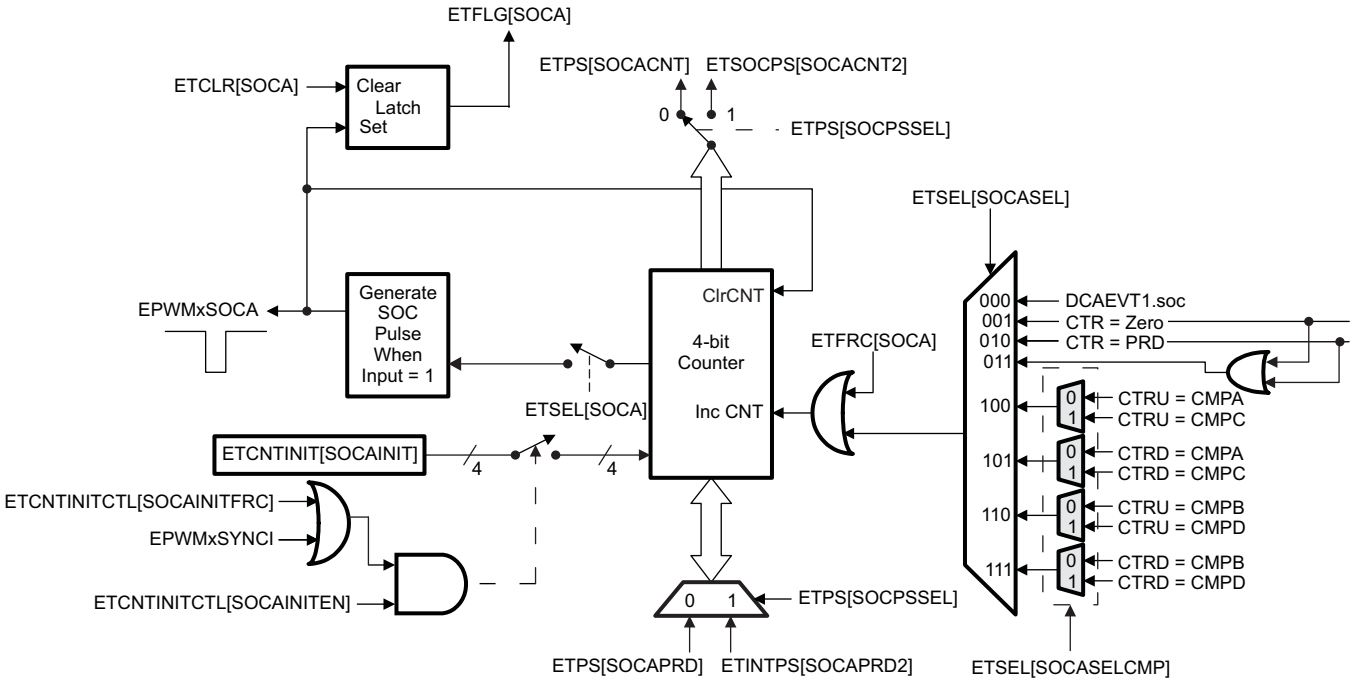


Figure 7-45 shows the operation of the event-trigger's start-of-conversion-A (SOCA) pulse generator. The enhancements include SOCASELCMP and SOCBSELCMP bit fields defined in the ETSEL register enable CMPC and CMPD events respectively to cause a start of conversion. The ETPS[SOCPSSEL] bit field determines whether SOCACNT2 and SOCAPRD2 take control or not. The ETPS[SOCACNT] counter and ETPS[SOCAPRD] period values behave similarly to the interrupt generator except that the pulses are continuously generated. That is, the pulse flag ETFLG[SOCA] is latched when a pulse is generated, but it does not stop further pulse generation. The enable/disable bit ETSEL[SOCAEN] stops pulse generation, but input events can still be counted until the period value is reached as with the interrupt generation logic. The event that will trigger an SOCA and SOCB pulse can be configured separately in the ETSEL[SOCASEL] and ETSEL[SOCBSEL] bits. The possible events are the same events that can be specified for the interrupt generation logic with the addition of the DCAEVT1.soc and DCBEVT1.soc event signals from the digital compare (DC) submodule. The SOCACNT2 initialization scheme is very similar to the interrupt generator with respective enable, value initialize and SYNC or software force options.

Copyright © 2012–2019, Texas Instruments Incorporated

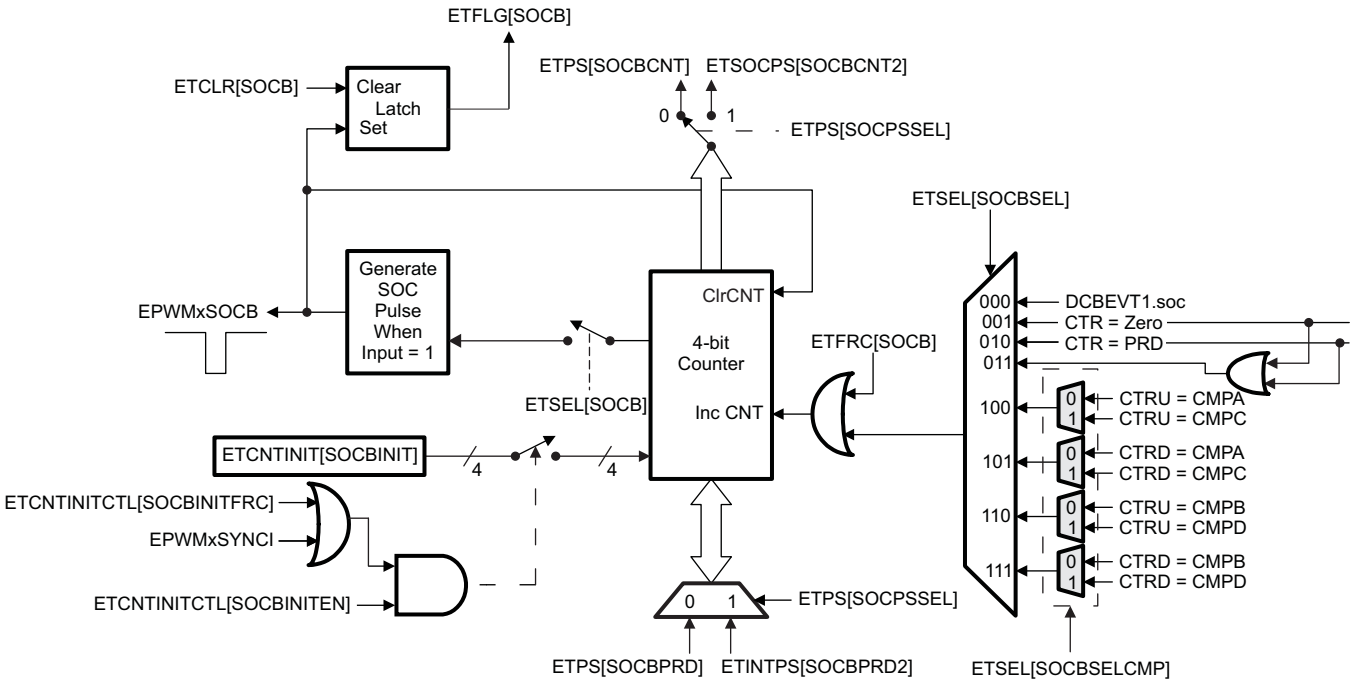
Figure 7-45. Event-Trigger SOCA Pulse Generator



A The DCAEVT1.soc signals are signals generated by the Digital compare (DC) submodule described later in Section 7.2.9.

Figure 7-46 shows the operation of the event-trigger's start-of-conversion-B (SOCB) pulse generator. The event-trigger's SOCB pulse generator operates the same way as the SOCA.

Figure 7-46. Event-Trigger SOCB Pulse Generator



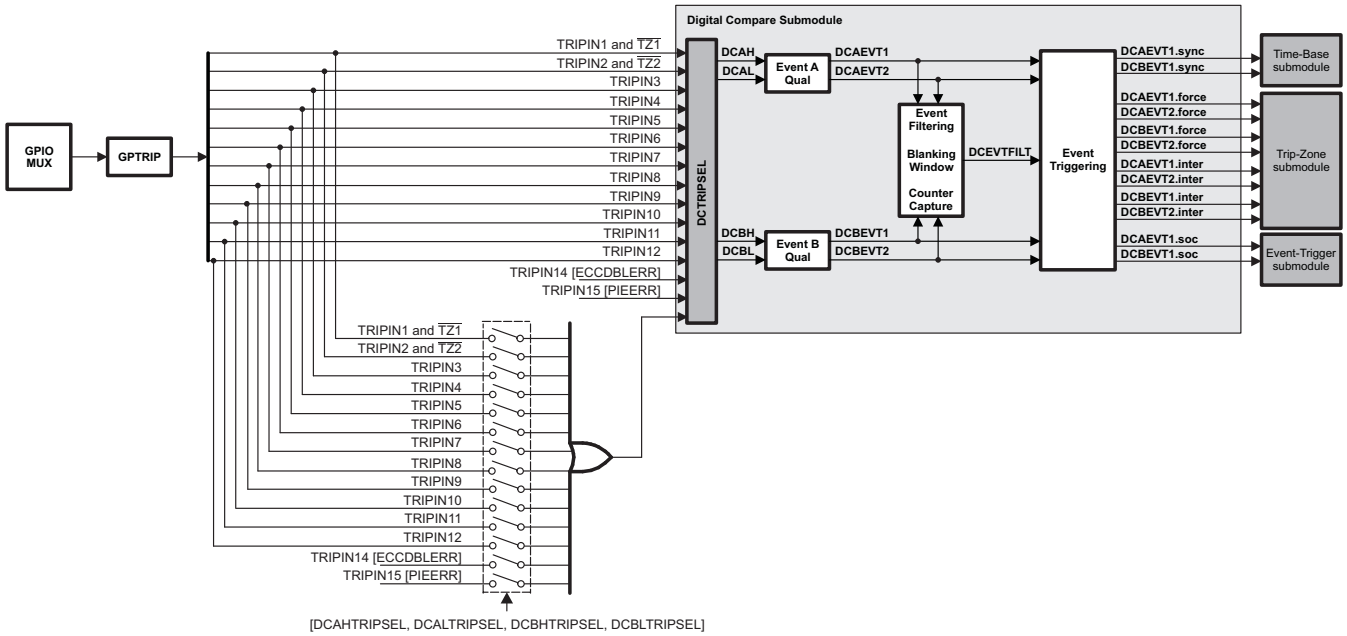
A The DCBEVT1.soc signals are signals generated by the Digital compare (DC) submodule in Section 7.2.9.

Copyright © 2012–2019, Texas Instruments Incorporated

7.2.9 Digital Compare (DC) Submodule

Figure 7-47 illustrates where the digital compare (DC) submodule signals interface to other submodules in the ePWM system.

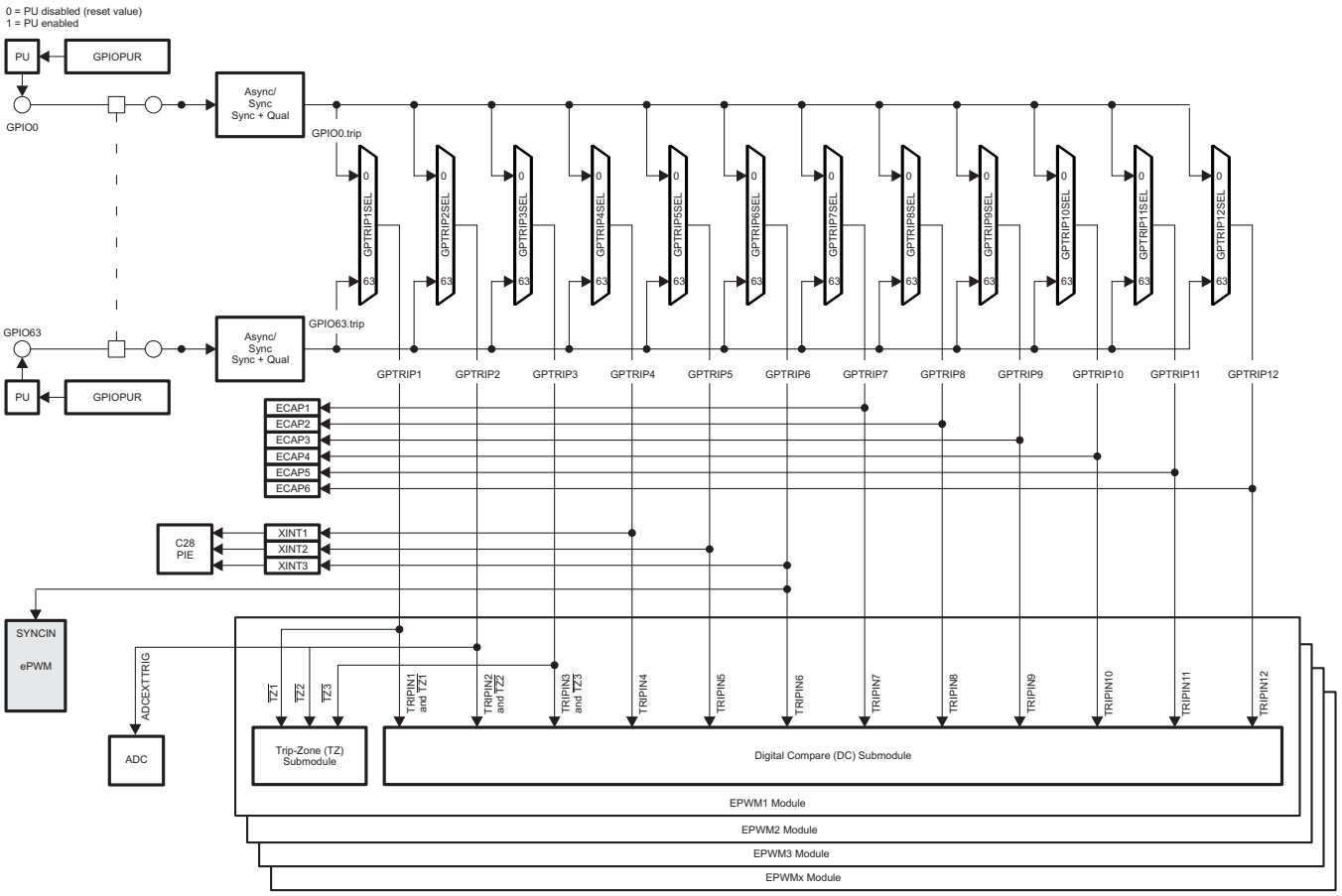
Figure 7-47. Digital-Compare Submodule High-Level Block Diagram



The ECAP input signals are sourced from the GPTRIP signals as shown in Figure 7-48.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-48. GPIO MUX-to-Trip Input Connectivity



On this device, any of the 64 GPIO pins can be flexibly mapped to be the trip-zone input and/or trip inputs to the trip-zone submodule and digital compare submodule. The GPIO Trip Input Select (GPTRIPxSEL) register defines which GPIO pins gets assigned to be the trip-zone inputs / trip inputs.

The digital compare (DC) submodule compares signals external to the ePWM module (for instance, COMPxOUT signals from the analog comparators) to directly generate PWM events/actions which then feed to the event-trigger, trip-zone, and time-base submodules. Additionally, blanking window functionality is supported to filter noise or unwanted pulses from the DC event signals.

NOTE: Enhanced trip input connectivity is specifically applicable to ePWM type 2. If a pin will be used as a C28 GPIO, the correct bits must be set in the GPIOCSEL register. This register is located in the M3 GPIO register space and more details can be found in the General-Purpose Input/Output (GPIO) chapter of this TRM. Once the user configures the respective pin as a C28 GPIO then the GPTRIPxSEL register which belongs to the GPTRIPx logic should be used to configure that specific pin as an external trip input. The user is responsible for driving correct state on the selected pin before enabling clock and configuring the trip input for the respective ePWM peripheral to avoid spurious latch of TRIP signal

7.2.9.1 Purpose of the Digital Compare Submodule

The key functions of the digital compare submodule are:

- Analog Comparator (COMP) module outputs fed though the GPTRIP logic externally using the GPIO peripheral, internal PIE, ECC error signals , TZ1, $\overline{TZ2}$, and $\overline{TZ3}$ inputs generate Digital Compare A High/Low (DCAH, DCAL) and Digital Compare B High/Low (DCBH, DCBL) signals.
- DCAH/L and DCBH/L signals trigger events which can then either be filtered or fed directly to the trip-

Copyright © 2012–2019, Texas Instruments Incorporated

zone, event-trigger, and time-base submodules to:
  - generate a trip zone interrupt
  - generate an ADC start of conversion
  - force an event
  - generate a synchronization event for synchronizing the ePWM module TBCTR.
- Event filtering (blanking window logic) can optionally blank the input signal to remove noise.

7.2.9.2 Enhanced Trip Action

In order to allow multiple comparators at a time to affect DCA/BEVTx events and trip actions, there is a OR logic to bring together ALL trip inputs (up to 15) from sources external to the ePWM module and feed into DCAH, DCAL, DCBH, and DCBL as “combinational input” using the DCTRIPSEL register. This is configured by writing appropriate value [Trip Combination input] to the DCAHCOMPSEL,DCALCOMPSEL, DCBHCOMPSEL, DCBLCOMPSEL bit fields in the DCTRIPSEL register.

The user has an discrete choice for which trip input to put through the combinational logic for Digital Compare A High/Low (DCAH, DCAL) and Digital Compare B High/Low (DCBH, DCBL) signals generation. This is acheived using the selection from DCAHTRIPSEL, DCALTRIPSEL, DCBHTRIPSEL and DCBLTRIPSEL register. The approprite bit when set indicates that Trip input is chosen for “combinational input” by the DCTRIPSEL register.

Apart from these options user can also make the external trip inputs which feed into the OR gate individually selectable and not go through the “combinational input” by using the DCTRIPSEL register.

7.2.9.3 Controlling and Monitoring the Digital Compare Submodule

The digital compare submodule operation is controlled and monitored through the following registers:

Table 7-24. Digital Compare Submodule Registers

Register Name	Address Offset	Shadowed	Description
TZDCSEL[1] [2]	0x13	No	Trip Zone Digital Compare Select Register
DCTRIPSEL[1]	0x30	No	Digital Compare Trip Select Register
DCACTL[1]	0x31	No	Digital Compare A Control Register
DCBCTL[1]	0x32	No	Digital Compare B Control Register
DCFCTL[1]	0x33	No	Digital Compare Filter Control Register
DCCAPCTL[1]	0x34	No	Digital Compare Capture Control Register
DCFOFFSET	0x35	Writes	Digital Compare Filter Offset Register
DCFOFFSETCNT	0x36	No	Digital Compare Filter Offset Counter Register
DCFWINDOW	0x37	No	Digital Compare Filter Window Register
DCFWINDOWCNT	0x38	No	Digital Compare Filter Window Counter Register
DCCAP	0x39	Yes	Digital Compare Counter Capture Register
DCAHTRIPSEL	0x4C	No	Digital Compare AH Trip Select
DCALTRIPSEL	0x4D	No	Digital Compare AL Trip Select
DCBHTRIPSEL	0x4E	No	Digital Compare BH Trip Select
DCBLTRIPSEL	0x4F	No	Digital Compare BL Trip Select

(1) These registers are EALLOW protected and can be modified only after executing the EALLOW instruction. For more information, see the device-specific version of the System Control and Interrupts Reference Guide.
(2) The TZDCSEL register is part of the trip-zone submodule but is mentioned again here because of its functional significance to the digital compare submodule.

Copyright © 2012–2019, Texas Instruments Incorporated

7.2.9.4 Operation Highlights of the Digital Compare Submodule

The following sections describe the operational highlights and configuration options for the digital compare submodule.

7.2.9.4.1 Digital Compare Events

As illustrated in Figure 7-47 earlier in this section, trip zone inputs ($\overline{TZ1}$, $\overline{TZ2}$, and $\overline{TZ3}$) and COMPxOUT signals from the analog comparator (COMP) module can be selected via the DCTRIPSEL bits to generate the Digital Compare A High and Low (DCAH/L) and Digital Compare B High and Low (DCBH/L) signals. Then, the configuration of the TZDCSEL register qualifies the actions on the selected DCAH/L and DCBH/L signals, which generate the DCAEVT1/2 and DCBEVT1/2 events (Event Qualification A and B).

NOTE: The $\overline{TZn}$ signals, when used as a DCEVT tripping functions, are treated as a normal input signal and can be defined to be active high or active low inputs. EPWM outputs are asynchronously tripped when either the $\overline{TZn}$, DCAEVTx.force, or DCBEVTx.force signals are active. For the condition to remain latched, a minimum of 3*TBCLK sync pulse width is required. If pulse width is < 3*TBCLK sync pulse width, the trip condition may or may not get latched by CBC or OST latches.

The DCAEVT1/2 and DCBEVT1/2 events can then be filtered to provide a filtered version of the event signals (DCEVTFILT) or the filtering can be bypassed. Filtering is discussed further in section 2.9.3.2. Either the DCAEVT1/2 and DCBEVT1/2 event signals or the filtered DCEVTFILT event signals can generate a force to the trip zone module, a TZ interrupt, an ADC SOC, or a PWM sync signal.

- **force signal:**

  DCAEVT1/2.force signals force trip zone conditions which either directly influence the output on the EPWMxA pin (via TZCTL[DCAEVT1 or DCAEVT2] configurations) or, if the DCAEVT1/2 signals are selected as one-shot or cycle-by-cycle trip sources (via the TZSEL register), the DCAEVT1/2.force signals can effect the trip action via the TZCTL[TZA] configuration. The DCBEVT1/2.force signals behaves similarly, but affect the EPWMxB output pin instead of the EPWMxA output pin.

  The priority of conflicting actions on the TZCTL register is as follows (highest priority overrides lower priority):

  Output EPWMxA: TZA (highest) -> DCAEVT1 -> DCAEVT2 (lowest)

  Output EPWMxB: TZB (highest) -> DCBEVT1 -> DCBEVT2 (lowest)

- **interrupt signal:**

  DCAEVT1/2.interrupt signals generate trip zone interrupts to the PIE. To enable the interrupt, the user must set the DCAEVT1, DCAEVT2, DCBEVT1, or DCBEVT2 bits in the TZEINT register. Once one of these events occurs, an EPWMxTZINT interrupt is triggered, and the corresponding bit in the TZCLR register must be set in order to clear the interrupt.

- **soc signal:**

  The DCAEVT1.soc signal interfaces with the event-trigger submodule and can be selected as an event which generates an ADC start-of-conversion-A (SOCA) pulse via the ETSEL[SOCASEL] bit. Likewise, the DCBEVT1.soc signal can be selected as an event which generates an ADC start-of-conversion-B (SOCB) pulse via the ETSEL[SOCBSEL] bit.

- **sync signal:**

  The DCAEVT1.sync and DCBEVT1.sync events are ORed with the EPWMxSYNCI input signal and the TBCTL[SWFSYNC] signal to generate a synchronization pulse to the time-base counter.

The diagrams below show how the DCAEVT1, DCAEVT2 or DCEVTFLT signals are processed to generate the digital compare A event force, interrupt, soc and sync signals.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-49. DCAEVT1 Event Triggering

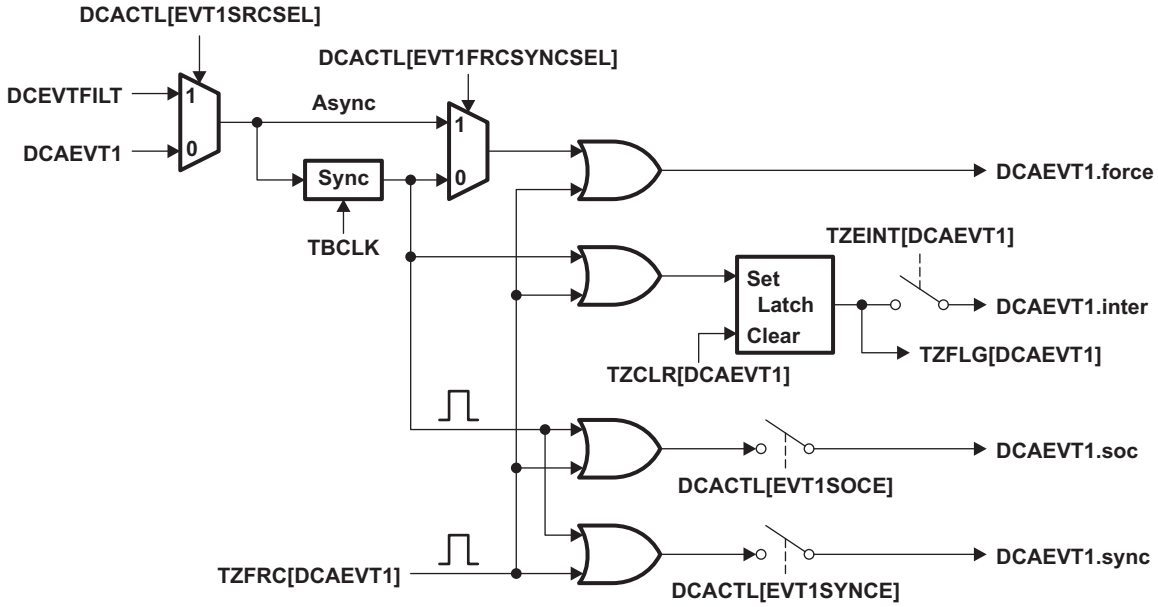
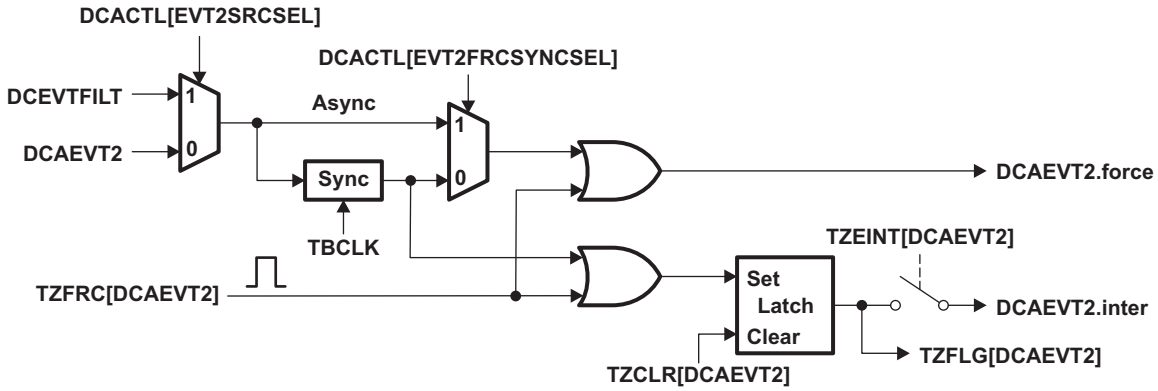


Figure 7-50. DCAEVT2 Event Triggering



Copyright © 2012–2019, Texas Instruments Incorporated

The diagrams below show how the DCBEVT1, DCBEVT2 or DCEVTFLT signals are processed to generate the digital compare B event force, interrupt, soc and sync signals.

Figure 7-51. DCBEVT1 Event Triggering

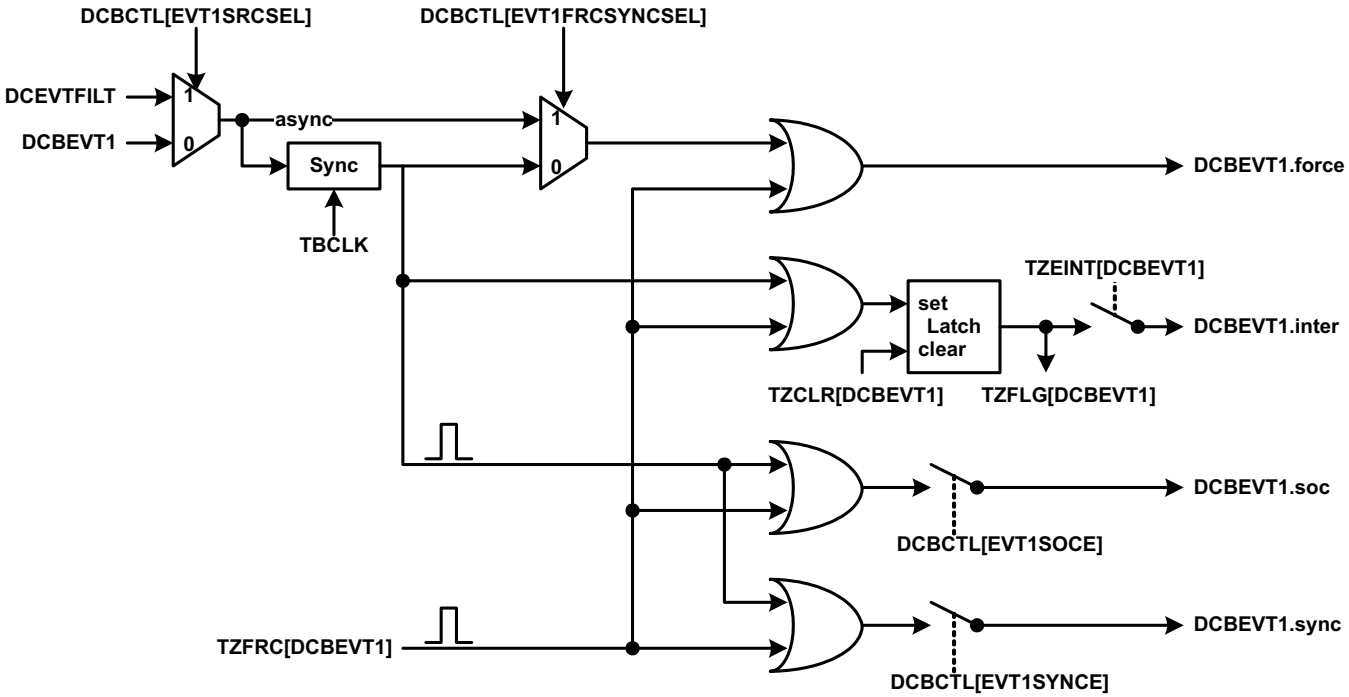
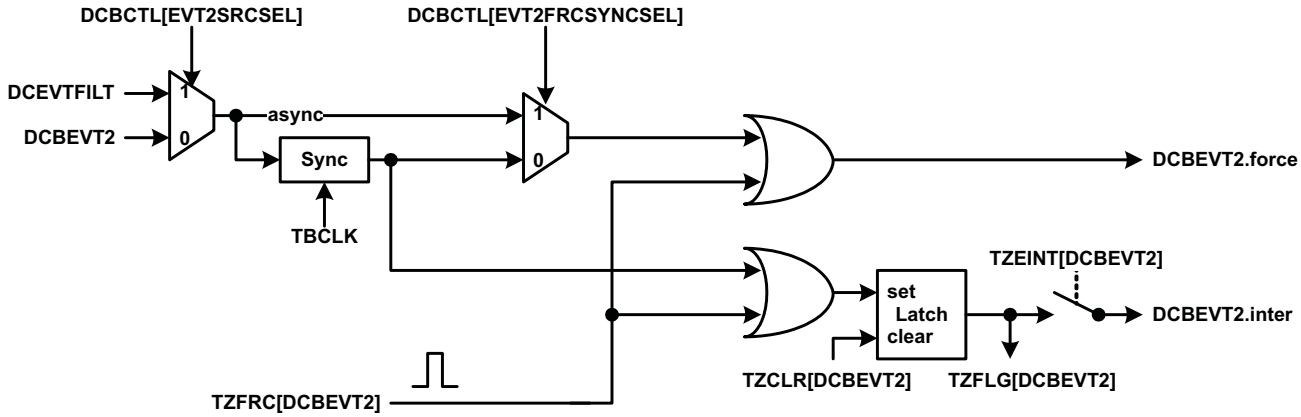


Figure 7-52. DCBEVT2 Event Triggering

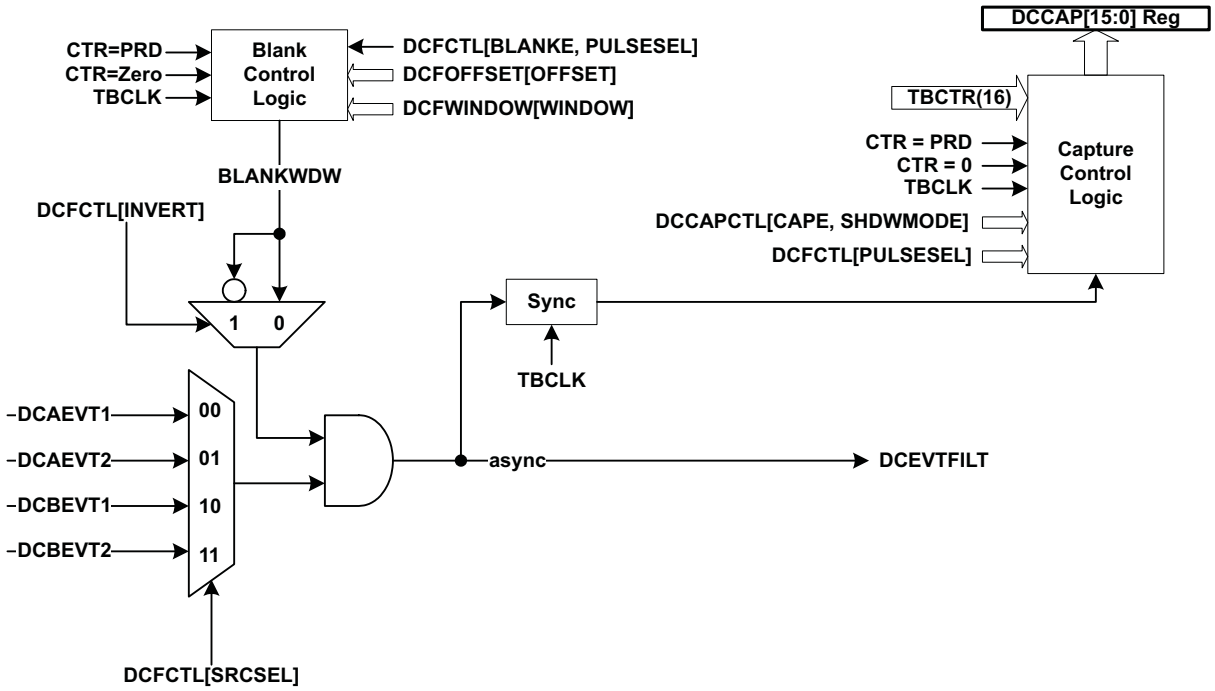


7.2.9.4.2 Event Filtering

The DCAEVT1/2 and DCBEVT1/2 events can be filtered via event filtering logic to remove noise by optionally blanking events for a certain period of time. This is useful for cases where the analog comparator outputs may be selected to trigger DCAEVT1/2 and DCBEVT1/2 events, and the blanking logic is used to filter out potential noise on the signal prior to tripping the PWM outputs or generating an interrupt or ADC start-of-conversion. The event filtering can also capture the TBCTR value of the trip event. The diagram below shows the details of the event filtering logic.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-53. Event Filtering

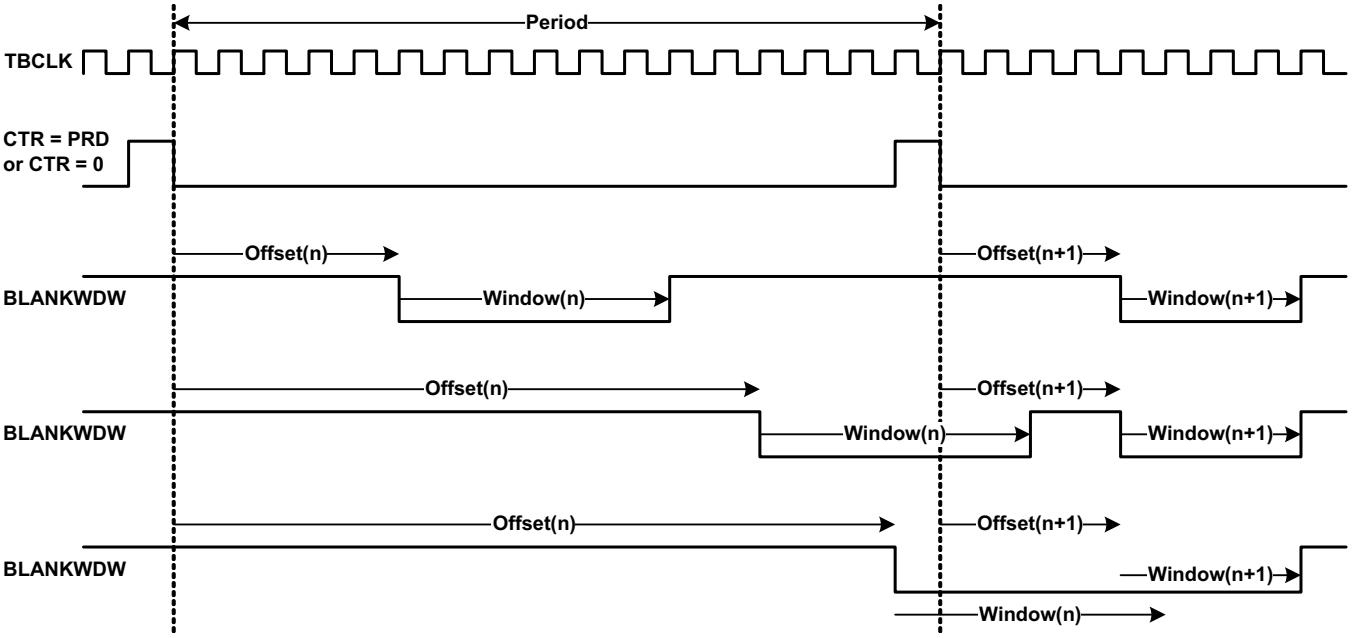


If the blanking logic is enabled, one of the digital compare events – DCAEVT1, DCAEVT2, DCBEVT1, DCBEVT2 – is selected for filtering. The blanking window, which filters out all event occurrences on the signal while it is active, will be aligned to either a CTR = PRD pulse or a CTR = 0 pulse (configured by the DCFCTL[PULSESEL] bits). An offset value in TBCLK counts is programmed into the DCFOFFSET register, which determines at what point after the CTR = PRD or CTR = 0 pulse the blanking window starts. The duration of the blanking window, in number of TBCLK counts after the offset counter expires, is written to the DCFWINDOW register by the application. During the blanking window, all events are ignored. Before and after the blanking window ends, events can generate soc, sync, interrupt, and force signals as before.

The diagram below illustrates several timing conditions for the offset and blanking window within an ePWM period. Notice that if the blanking window crosses the CTR = 0 or CTR = PRD boundary, the next window still starts at the same offset value after the CTR = 0 or CTR = PRD pulse.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-54. Blanking Window Timing Diagram



Copyright © 2012–2019, Texas Instruments Incorporated

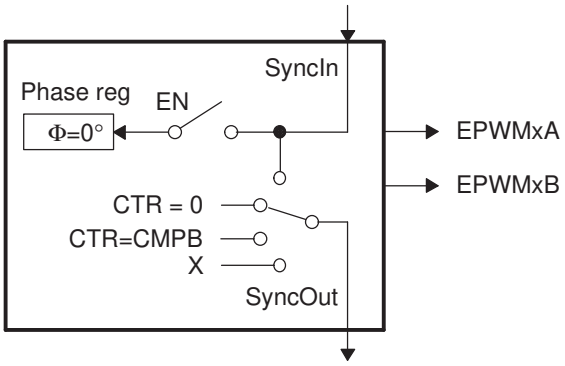
7.3 Applications to Power Topologies

An ePWM module has all the local resources necessary to operate completely as a standalone module or to operate in synchronization with other identical ePWM modules.

7.3.1 Overview of Multiple Modules

Previously in this user's guide, all discussions have described the operation of a single module. To facilitate the understanding of multiple modules working together in a system, the ePWM module described in reference is represented by the more simplified block diagram shown in Figure 7-55. This simplified ePWM block shows only the key resources needed to explain how a multiswitch power topology is controlled with multiple ePWM modules working together.

Figure 7-55. Simplified ePWM Module



7.3.2 Key Configuration Capabilities

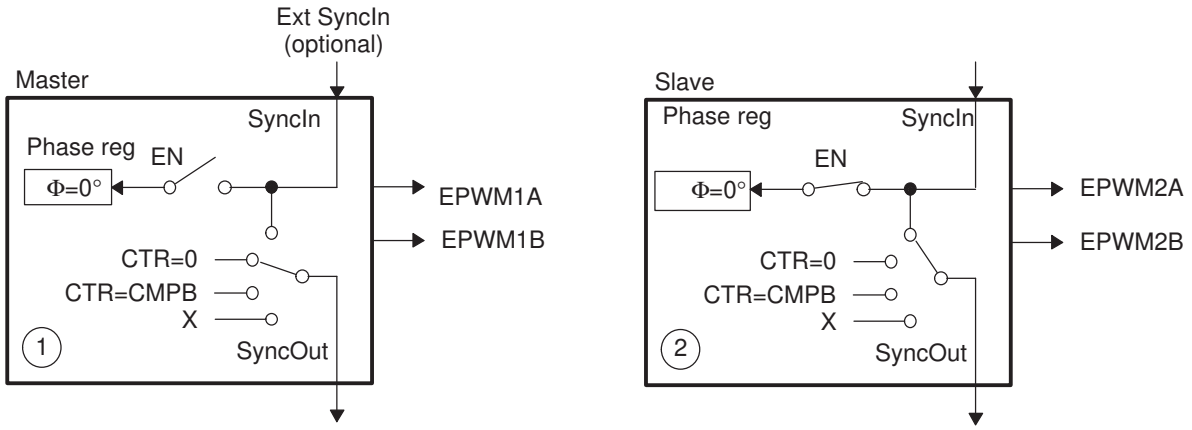
The key configuration choices available to each module are as follows:

- Options for SyncIn
  - Load own counter with phase register on an incoming sync strobe—enable (EN) switch closed
  - Do nothing or ignore incoming sync strobe—enable switch open
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)
- Options for SyncOut
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)

For each choice of SyncOut, a module may also choose to load its own counter with a new phase value on a SyncIn strobe input or choose to ignore it, i.e., via the enable switch. Although various combinations are possible, the two most common—master module and slave module modes—are shown in Figure 7-56.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-56. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave

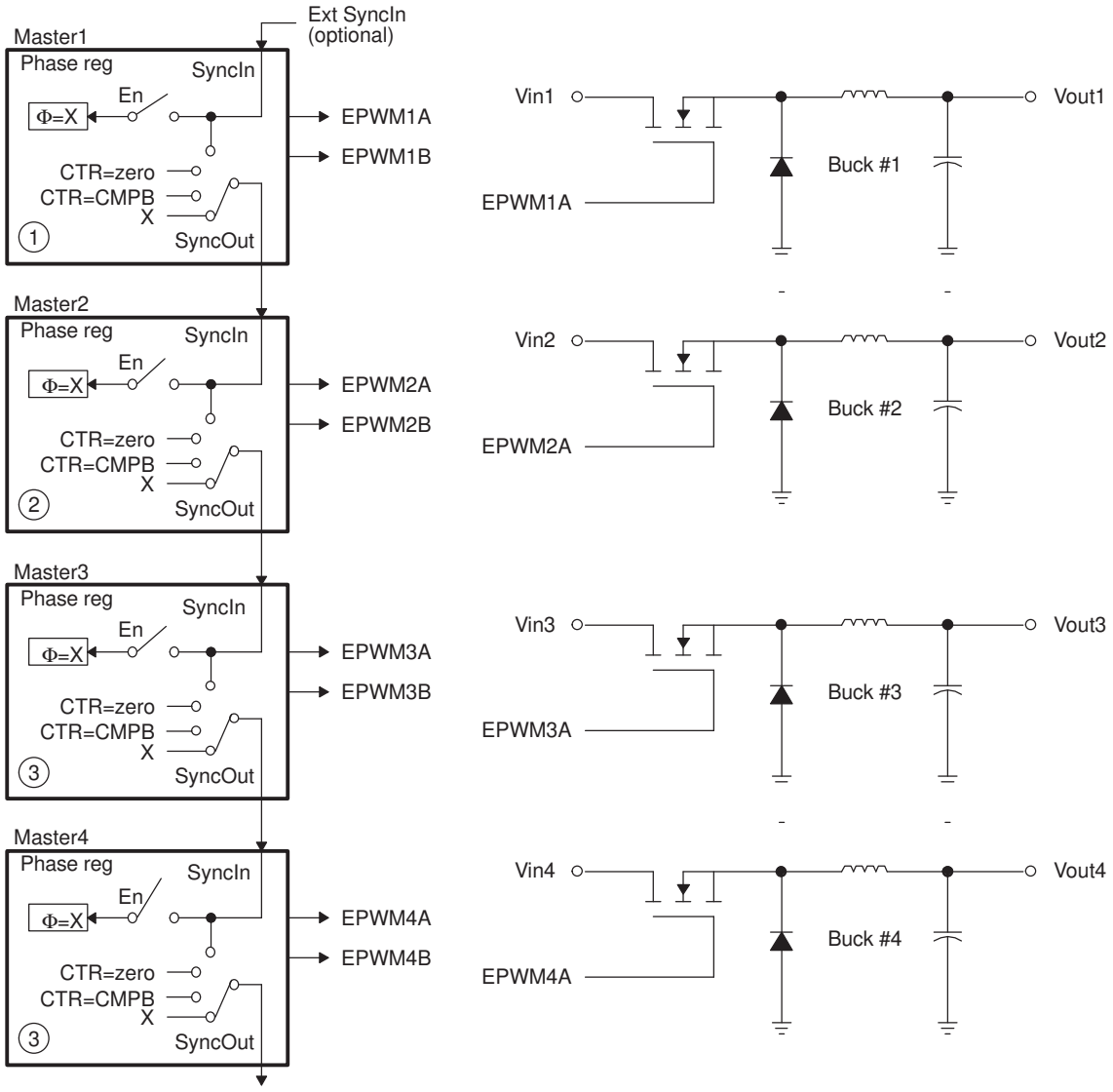


7.3.3 Controlling Multiple Buck Converters With Independent Frequencies

One of the simplest power converter topologies is the buck. A single ePWM module configured as a master can control two buck stages with the same PWM frequency. If independent frequency control is required for each buck converter, then one ePWM module must be allocated for each converter stage. Figure 7-57 shows four buck stages, each running at independent frequencies. In this case, all four ePWM modules are configured as Masters and no synchronization is used. Figure 7-58 shows the waveforms generated by the setup shown in Figure 7-57; note that only three waveforms are shown, although there are four stages.

Copyright © 2012–2019, Texas Instruments Incorporated

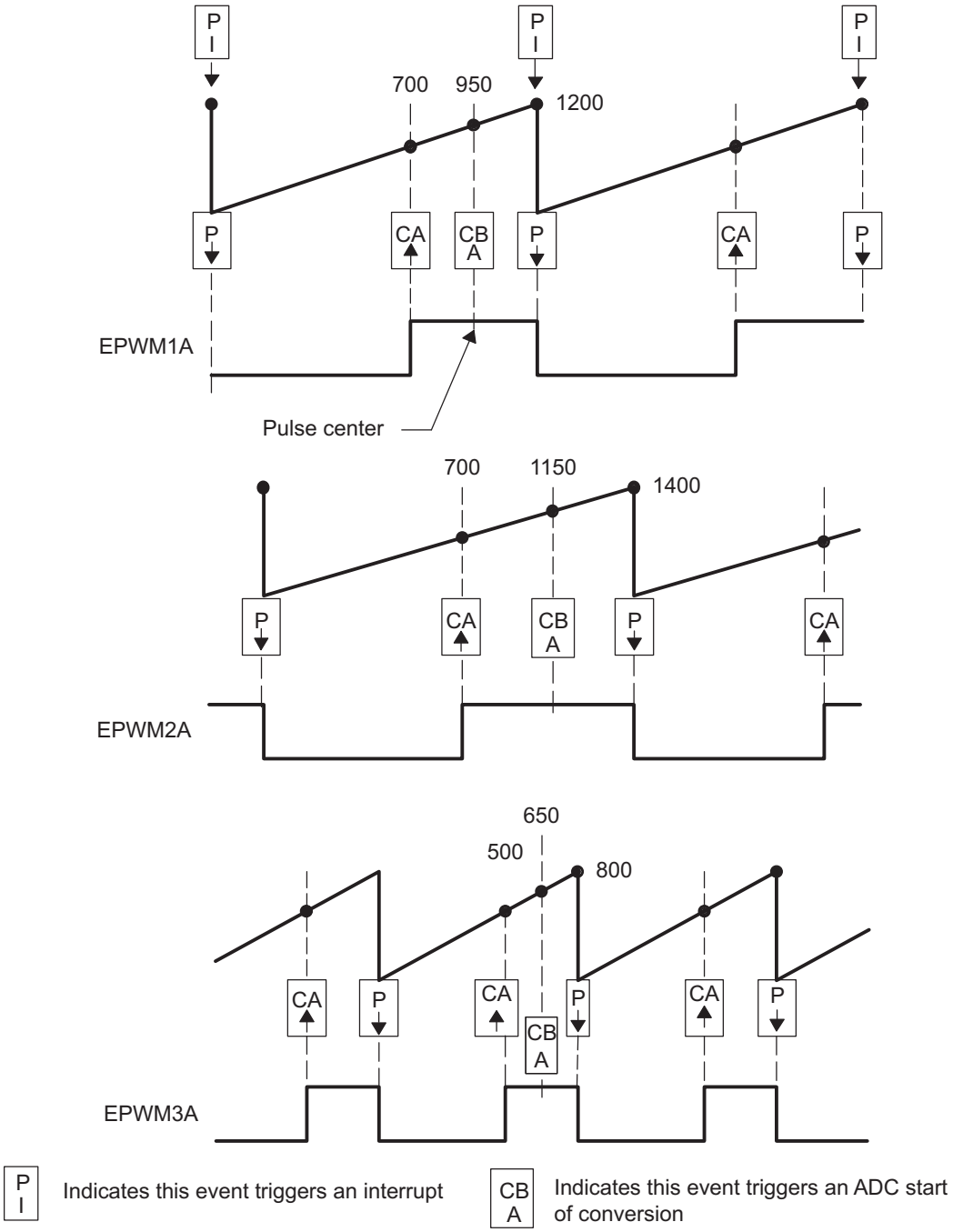
Figure 7-57. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$



NOTE: Θ = X indicates value in phase register is a "don't care"

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-58. Buck Waveforms for Figure 7-57 (Note: Only three bucks shown here)



Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-8. Configuration for Example in Figure 7-58

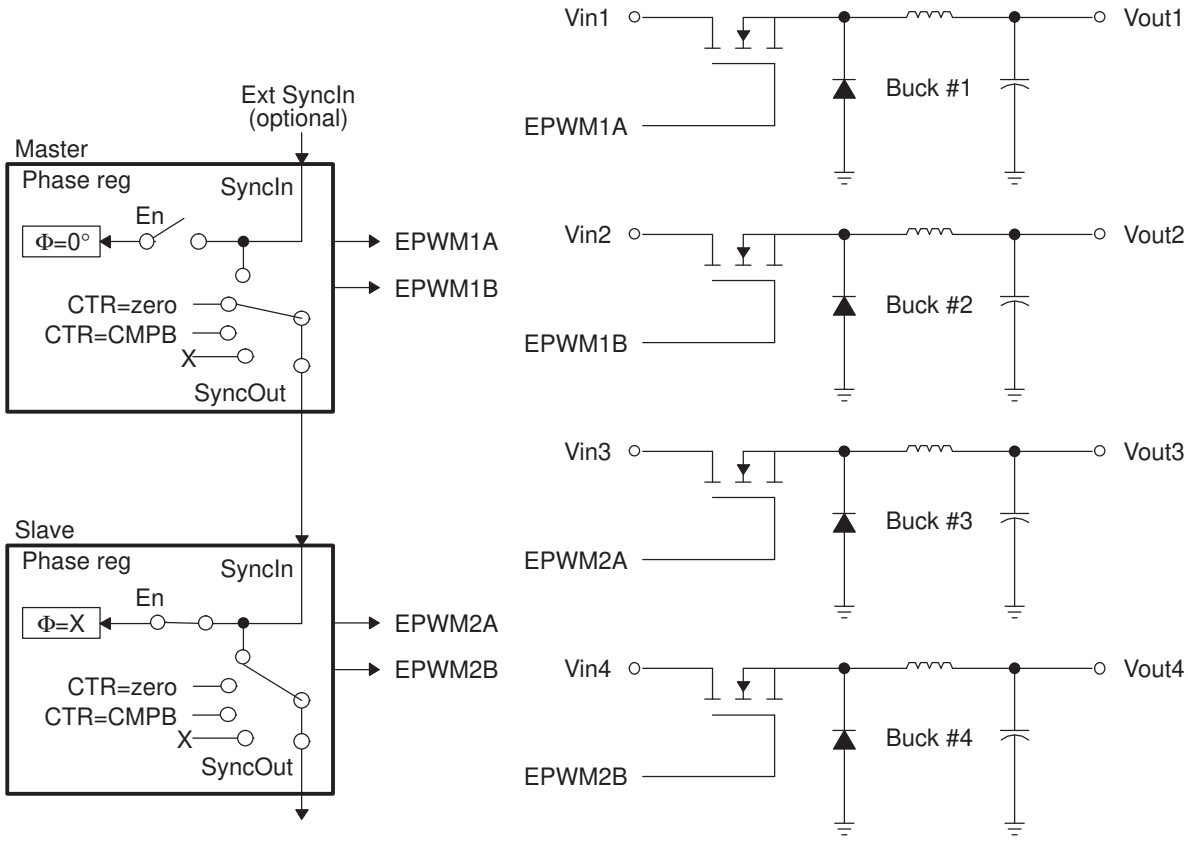
```
//=====================================================================
// (Note: code for only 3 modules shown)
// Initialization Time
//=======================
// EPWM Module 1 config
   EPwm1Regs.TBPRD = 1200;                         // Period = 1201 TBCLK counts
   EPwm1Regs.TBPHS.half.TBPHS = 0;                 // Set Phase register to zero
   EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;      // Asymmetrical mode
   EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Phase loading disabled
   EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
   EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
   EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
   EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
   EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
   EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
   EPwm1Regs.AQCTLA.bit.PRD = AQ_CLEAR;
   EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
// EPWM Module 2 config
   EPwm2Regs.TBPRD = 1400;                         // Period = 1401 TBCLK counts
   EPwm2Regs.TBPHS.half.TBPHS = 0;                 // Set Phase register to zero
   EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;      // Asymmetrical mode
   EPwm2Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Phase loading disabled
   EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
   EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
   EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
   EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
   EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
   EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
   EPwm2Regs.AQCTLA.bit.PRD = AQ_CLEAR;
   EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;
// EPWM Module 3 config
   EPwm3Regs.TBPRD = 800;                           // Period = 801 TBCLK counts
   EPwm3Regs.TBPHS.half.TBPHS = 0;                  // Set Phase register to zero
   EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
   EPwm3Regs.TBCTL.bit.PHSEN = TB_DISABLE;          // Phase loading disabled
   EPwm3Regs.TBCTL.bit.PRDLD = TB_SHADOW;
   EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
   EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
   EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
   EPwm3Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;    // load on CTR=Zero
   EPwm3Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;    // load on CTR=Zero
   EPwm3Regs.AQCTLA.bit.PRD = AQ_CLEAR;
   EPwm3Regs.AQCTLA.bit.CAU = AQ_SET;
//
// Run Time (Note: Example execution of one run-time instant)
//=====================================================
   EPwm1Regs.CMPA.half.CMPA = 700;                 // adjust duty for output EPWM1A
   EPwm2Regs.CMPA.half.CMPA = 700;                 // adjust duty for output EPWM2A
   EPwm3Regs.CMPA.half.CMPA = 500;                 // adjust duty for output EPWM3A
```


Copyright © 2012–2019, Texas Instruments Incorporated

7.3.4 Controlling Multiple Buck Converters With Same Frequencies

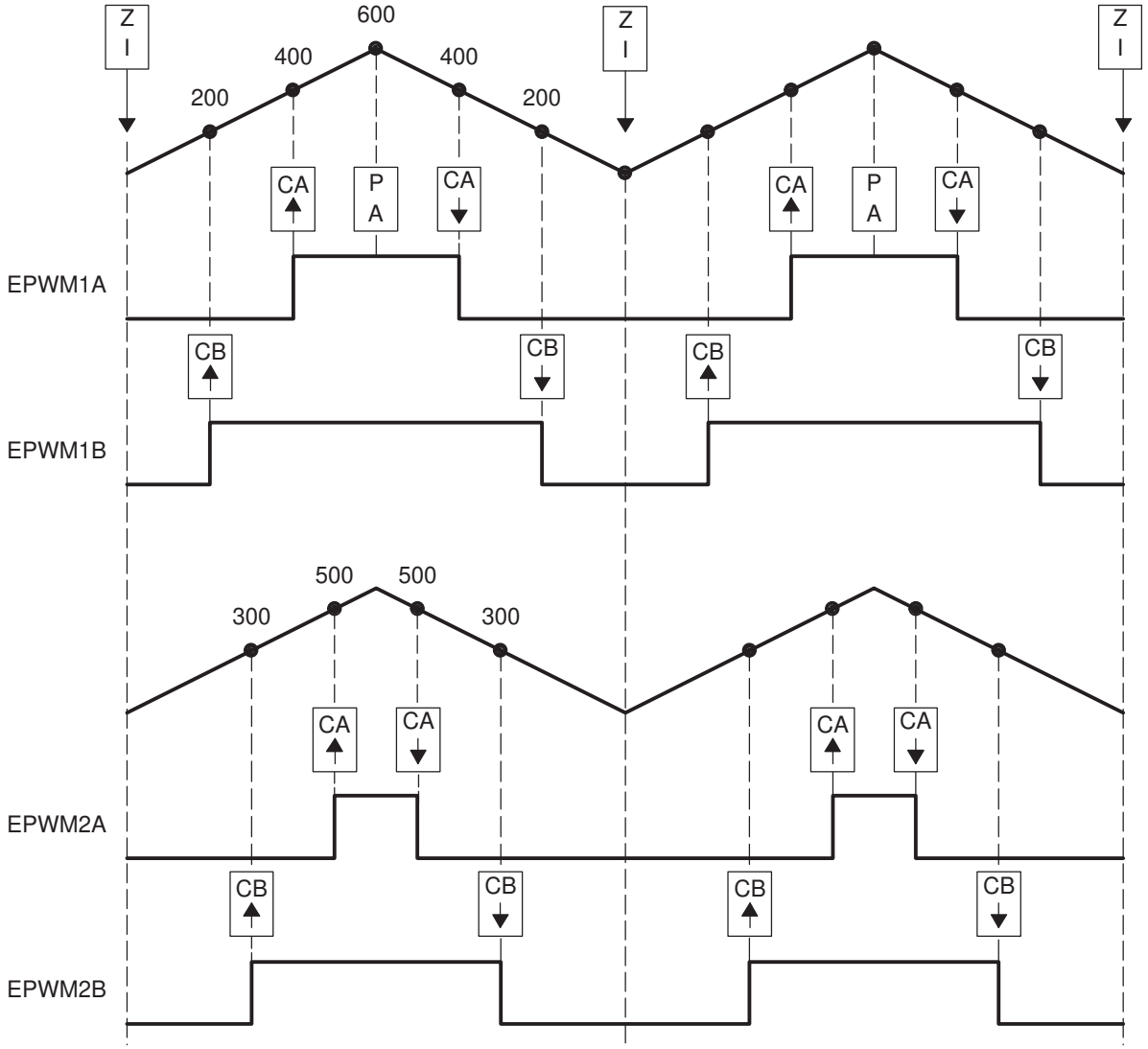
If synchronization is a requirement, ePWM module 2 can be configured as a slave and can operate at integer multiple (N) frequencies of module 1. The sync signal from master to slave ensures these modules remain locked. Figure 7-59 shows such a configuration; Figure 7-60 shows the waveforms generated by the configuration.

Figure 7-59. Control of Four Buck Stages. (Note: F_{PWM2} = N x F_{PWM1})



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-60. Buck Waveforms for Figure 7-59 (Note: $F_{PWM2} = F_{PWM1}$)



Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-9. Code Snippet for Configuration in Figure 7-59

```
//=======================
// EPWM Module 1 config
     EPwm1Regs.TBPRD = 600;                        // Period = 1200 TBCLK counts
     EPwm1Regs.TBPHS.half.TBPHS = 0;               // Set Phase register to zero
     EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
     EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;       // Master module
     EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;   // Sync down-stream module
     EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;            // set actions for EPWM1A
     EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
     EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;            // set actions for EPWM1B
     EPwm1Regs.AQCTLB.bit.CBD = AQ_CLEAR;
// EPWM Module 2 config
     EPwm2Regs.TBPRD = 600;                        // Period = 1200 TBCLK counts
     EPwm2Regs.TBPHS.half.TBPHS = 0;               // Set Phase register to zero
     EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
     EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;        // Slave module
     EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;    // sync flow-through
     EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;            // set actions for EPWM2A
     EPwm2Regs.AQCTLA.bit.CAD = AQ_CLEAR;
     EPwm2Regs.AQCTLB.bit.CBU = AQ_SET;            // set actions for EPWM2B
     EPwm2Regs.AQCTLB.bit.CBD = AQ_CLEAR;
//
// Run Time (Note: Example execution of one run-time instance)
//=====================================================
    EPwm1Regs.CMPA.half.CMPA = 400;               // adjust duty for output EPWM1A
    EPwm1Regs.CMPB = 200;                         // adjust duty for output EPWM1B
    EPwm2Regs.CMPA.half.CMPA = 500;               // adjust duty for output EPWM2A
    EPwm2Regs.CMPB = 300;                         // adjust duty for output EPWM2B
```

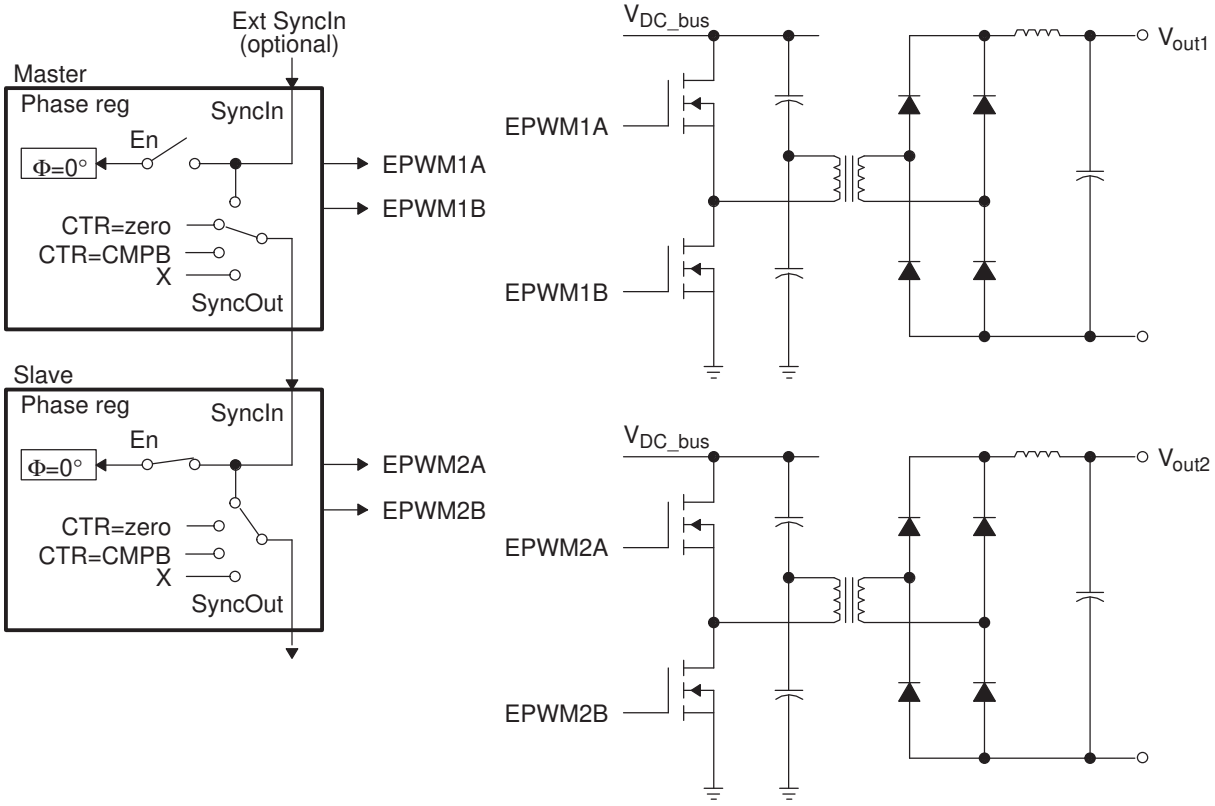

Copyright © 2012–2019, Texas Instruments Incorporated

7.3.5 Controlling Multiple Half H-Bridge (HHB) Converters

Topologies that require control of multiple switching elements can also be addressed with these same ePWM modules. It is possible to control a Half-H bridge stage with a single ePWM module. This control can be extended to multiple stages. Figure 7-61 shows control of two synchronized Half-H bridge stages where stage 2 can operate at integer multiple (N) frequencies of stage 1. Figure 7-62 shows the waveforms generated by the configuration shown in Figure 7-61.

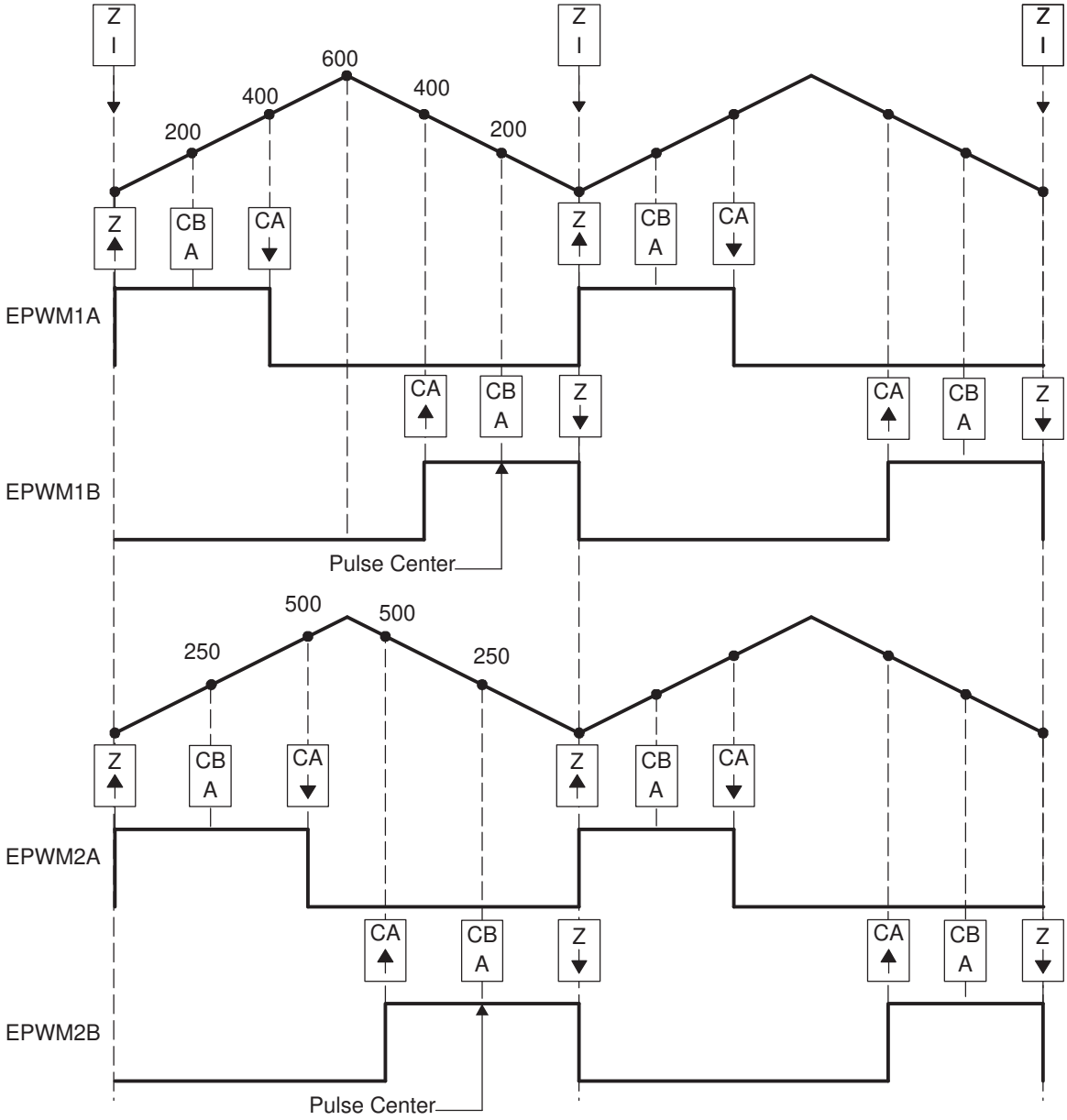
Module 2 (slave) is configured for Sync flow-through; if required, this configuration allows for a third Half-H bridge to be controlled by PWM module 3 and also, most importantly, to remain in synchronization with master module 1.

Figure 7-61. Control of Two Half-H Bridge Stages (F_{PWM2} = N x F_{PWM1})



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-62. Half-H Bridge Waveforms for Figure 7-61 (Note: Here $F_{PWM2} = F_{PWM1}$)



Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-10. Code Snippet for Configuration in Figure 7-61

```
//=====================================================================
// Config
//=====================================================================
// Initialization Time
//=======================
// EPWM Module 1 config
     EPwm1Regs.TBPRD = 600;                        // Period = 1200 TBCLK counts
     EPwm1Regs.TBPHS.half.TBPHS = 0;               // Set Phase register to zero
     EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
     EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;       // Master module
     EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;   // Sync down-stream module
     EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;            // set actions for EPWM1A
     EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
     EPwm1Regs.AQCTLB.bit.ZRO = AQ_CLEAR;          // set actions for EPWM1B
     EPwm1Regs.AQCTLB.bit.CAD = AQ_SET;
// EPWM Module 2 config
     EPwm2Regs.TBPRD = 600;                        // Period = 1200 TBCLK counts
     EPwm2Regs.TBPHS.half.TBPHS = 0;               // Set Phase register to zero
     EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
     EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;        // Slave module
     EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;    // sync flow-through
     EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm2Regs.AQCTLA.bit.ZRO = AQ_SET;            // set actions for EPWM1A
     EPwm2Regs.AQCTLA.bit.CAU = AQ_CLEAR;
     EPwm2Regs.AQCTLB.bit.ZRO = AQ_CLEAR;          // set actions for EPWM1B
     EPwm2Regs.AQCTLB.bit.CAD = AQ_SET;
//=========================================================
     EPwm1Regs.CMPA.half.CMPA = 400;               // adjust duty for output EPWM1A & EPWM1B

     EPwm1Regs.CMPB = 200;                         // adjust point-in-time for ADCSOC trigger
     EPwm2Regs.CMPA.half.CMPA = 500;               // adjust duty for output EPWM2A & EPWM2B
     EPwm2Regs.CMPB = 250;                         // adjust point-in-time for ADCSOC trigger
```

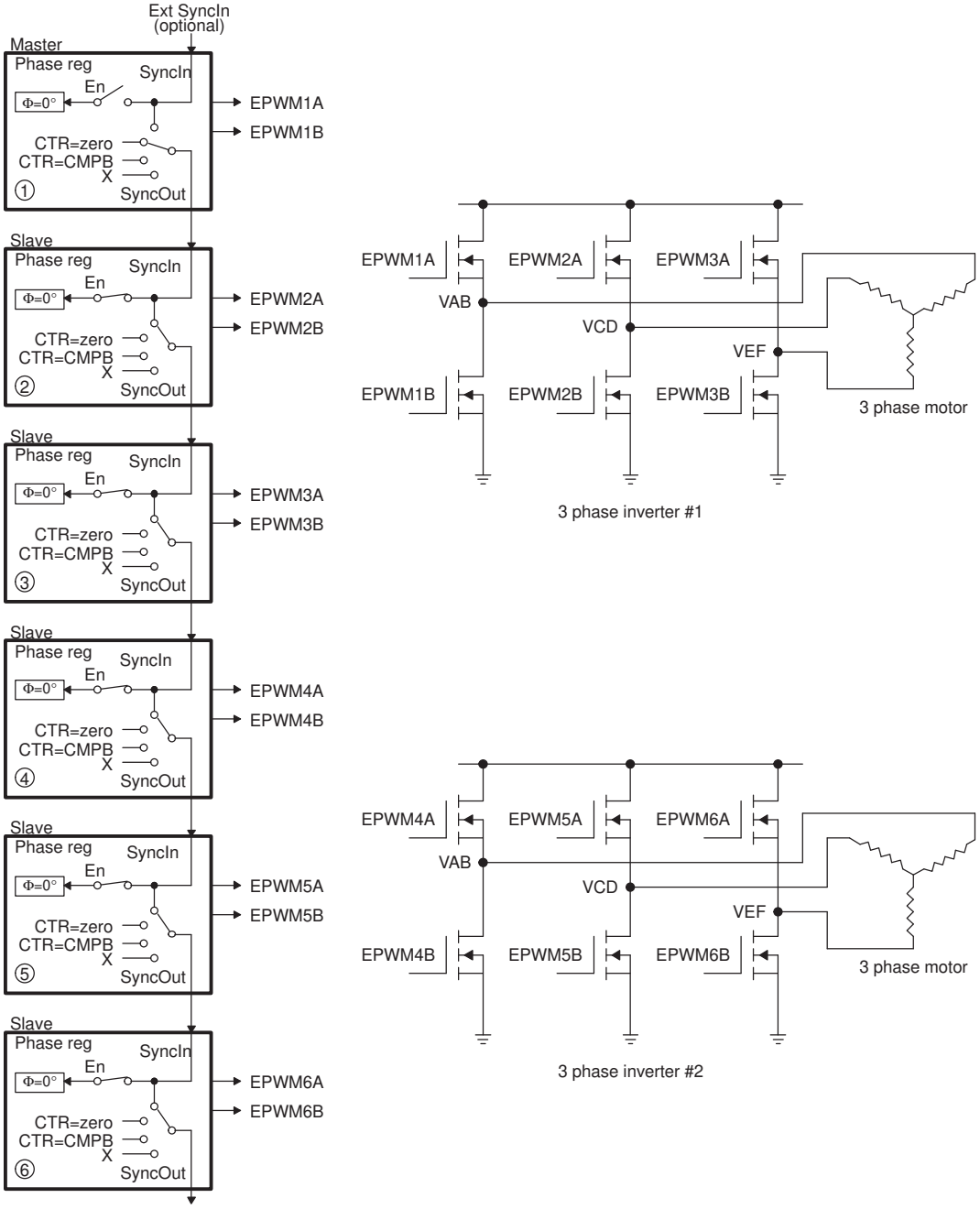
7.3.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM)

The idea of multiple modules controlling a single power stage can be extended to the 3-phase Inverter case. In such a case, six switching elements can be controlled using three PWM modules, one for each leg of the inverter. Each leg must switch at the same frequency and all legs must be synchronized. A master + two slaves configuration can easily address this requirement. Figure 7-63 shows how six PWM modules can control two independent 3-phase Inverters; each running a motor.

As in the cases shown in the previous sections, we have a choice of running each inverter at a different frequency (module 1 and module 4 are masters as in Figure 7-63), or both inverters can be synchronized by using one master (module 1) and five slaves. In this case, the frequency of modules 4, 5, and 6 (all equal) can be integer multiples of the frequency for modules 1, 2, 3 (also all equal).

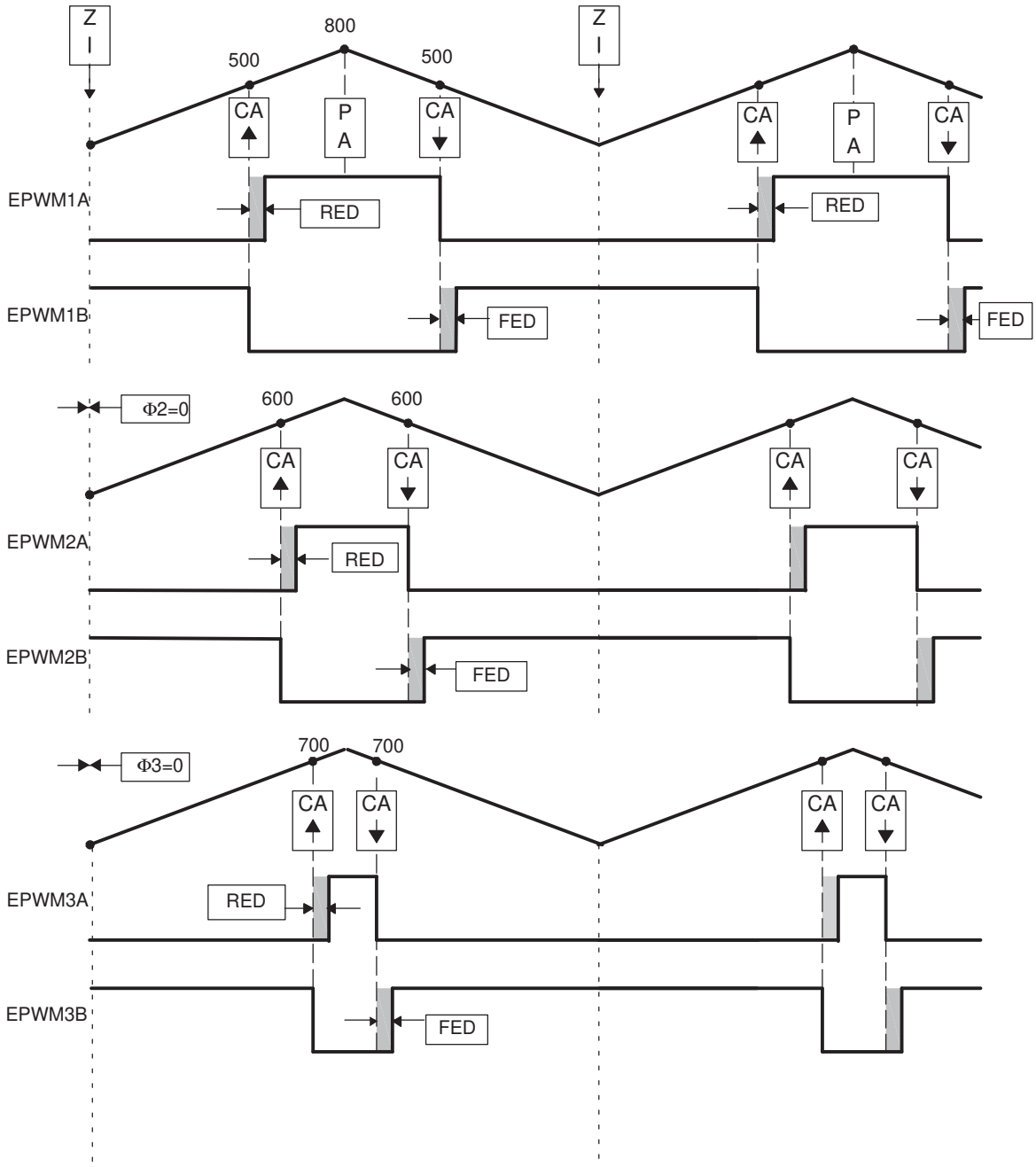
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-63. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-64. 3-Phase Inverter Waveforms for Figure 7-63 (Only One Inverter Shown)



Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-11. Code Snippet for Configuration in Figure 7-63

```
//=====================================================================
// Configuration
//=====================================================================
// Initialization Time
//=======================// EPWM Module 1 config
     EPwm1Regs.TBPRD = 800;                           // Period = 1600 TBCLK counts
     EPwm1Regs.TBPHS.half.TBPHS = 0;                  // Set Phase register to zero
     EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;   // Symmetrical mode
     EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;          // Master module
     EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;      // Sync down-stream module
     EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;    // load on CTR=Zero
     EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;    // load on CTR=Zero
     EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;               // set actions for EPWM1A
     EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
     EPwm1Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;   // enable Dead-band module
     EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;        // Active Hi complementary
     EPwm1Regs.DBFED = 50;                            // FED = 50 TBCLKs
     EPwm1Regs.DBRED = 50;                            // RED = 50 TBCLKs
// EPWM Module 2 config
     EPwm2Regs.TBPRD = 800;                           // Period = 1600 TBCLK counts
     EPwm2Regs.TBPHS.half.TBPHS = 0;                  // Set Phase register to zero
     EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;   // Symmetrical mode
     EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;           // Slave module
     EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;       // sync flow-through
     EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;    // load on CTR=Zero
     EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;    // load on CTR=Zero
     EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;               // set actions for EPWM2A
     EPwm2Regs.AQCTLA.bit.CAD = AQ_CLEAR;
     EPwm2Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;   // enable Dead-band module
     EPwm2Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;        // Active Hi complementary
     EPwm2Regs.DBFED = 50;                            // FED = 50 TBCLKs
     EPwm2Regs.DBRED = 50;                            // RED = 50 TBCLKs
// EPWM Module 3 config
     EPwm3Regs.TBPRD = 800;                           // Period = 1600 TBCLK counts
     EPwm3Regs.TBPHS.half.TBPHS = 0;                  // Set Phase register to zero
     EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;   // Symmetrical mode
     EPwm3Regs.TBCTL.bit.PHSEN = TB_ENABLE;           // Slave module
     EPwm3Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;       // sync flow-through
     EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm3Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;    // load on CTR=Zero
     EPwm3Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;    // load on CTR=Zero
     EPwm3Regs.AQCTLA.bit.CAU = AQ_SET;               // set actions for EPWM3A
     EPwm3Regs.AQCTLA.bit.CAD = AQ_CLEAR;
     EPwm3Regs.DBCTL.bitMODE = DB_FULL_ENABLE;       // enable Dead-band module
     EPwm3Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;        // Active Hi complementary
     EPwm3Regs.DBFED = 50;                            // FED = 50 TBCLKs
     EPwm3Regs.DBRED = 50;                            // RED = 50 TBCLKs
// Run Time (Note: Example execution of one run-time instant)
//=====================================================
     EPwm1Regs.CMPA.half.CMPA = 500;                  // adjust duty for output EPWM1A
     EPwm2Regs.CMPA.half.CMPA = 600;                  // adjust duty for output EPWM2A
     EPwm3Regs.CMPA.half.CMPA = 700;                  // adjust duty for output EPWM3A
```


Copyright © 2012–2019, Texas Instruments Incorporated

7.3.7 Practical Applications Using Phase Control Between PWM Modules

So far, none of the examples have made use of the phase register (TBPHS). It has either been set to zero or its value has been a don't care. However, by programming appropriate values into TBPHS, multiple PWM modules can address another class of power topologies that rely on phase relationship between legs (or stages) for correct operation. As described in the TB module section, a PWM module can be configured to allow a SyncIn pulse to cause the TBPHS register to be loaded into the TBCTR register. To illustrate this concept, Figure 7-65 shows a master and slave module with a phase relationship of 120°, i.e., the slave leads the master.

Figure 7-65. Configuring Two PWM Modules for Phase Control

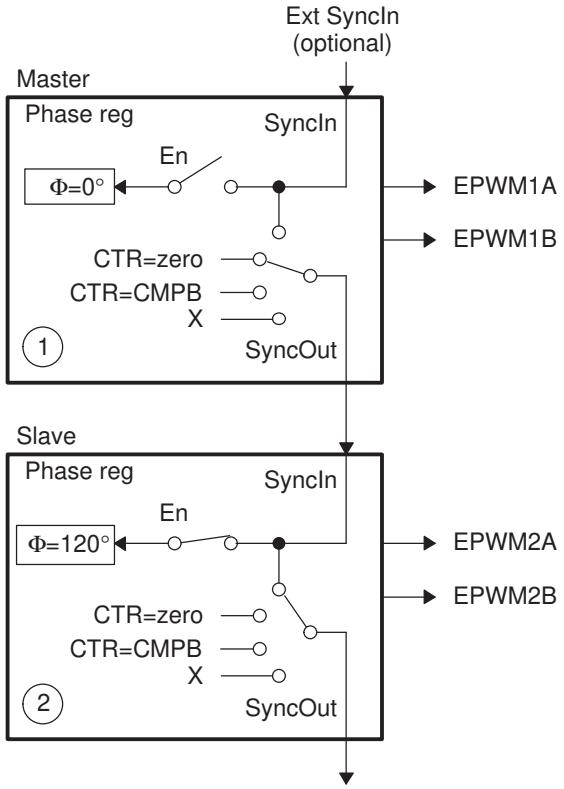
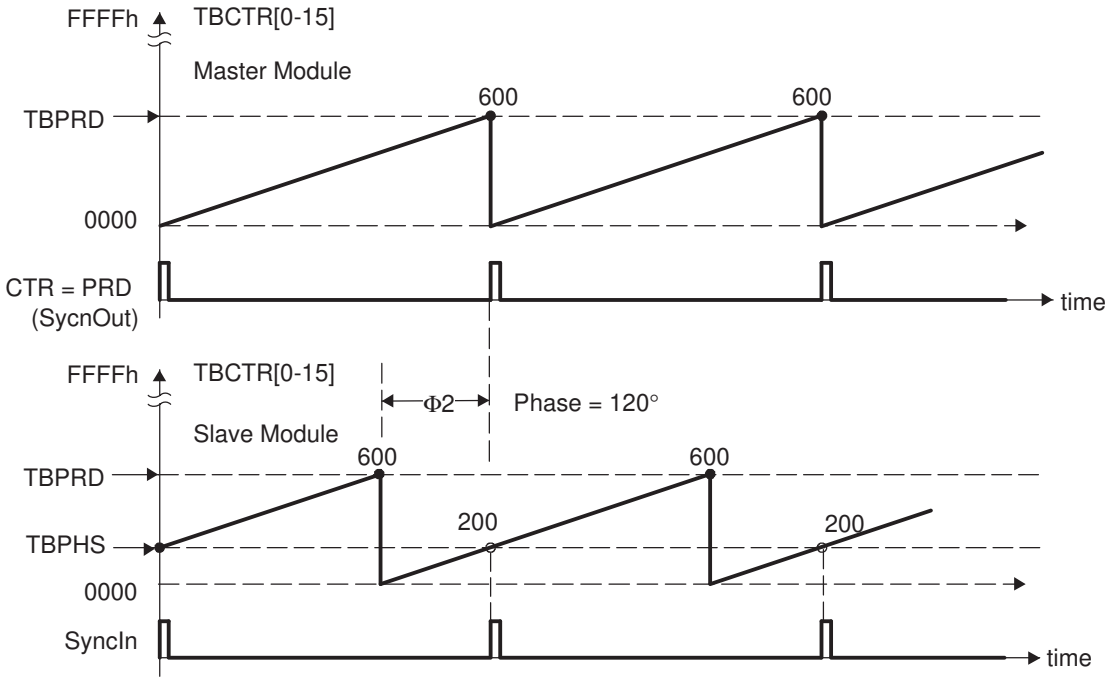


Figure 7-66 shows the associated timing waveforms for this configuration. Here, TBPRD = 600 for both master and slave. For the slave, TBPHS = 200 (i.e., 200/600 X 360° = 120°). Whenever the master generates a SyncIn pulse (CTR = PRD), the value of TBPHS = 200 is loaded into the slave TBCTR register so the slave time-base is always leading the master's time-base by 120°.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-66. Timing Waveforms Associated With Phase Control Between 2 Modules



7.3.8 Controlling a 3-Phase Interleaved DC/DC Converter

A popular power topology that makes use of phase-offset between modules is shown in Figure 7-67. This system uses three PWM modules, with module 1 configured as the master. To work, the phase relationship between adjacent modules must be F = 120°. This is achieved by setting the slave TBPHS registers 2 and 3 with values of 1/3 and 2/3 of the period value, respectively. For example, if the period register is loaded with a value of 600 counts, then TBPHS (slave 2) = 200 and TBPHS (slave 3) = 400. Both slave modules are synchronized to the master 1 module.

This concept can be extended to four or more phases, by setting the TBPHS values appropriately. The following formula gives the TBPHS values for N phases:

TBPHS(N,M) = (TBPRD/N) x (—1)

Where:

N = number of phases

M = PWM module number

For example, for the 3-phase case (N=3), TBPRD = 600,

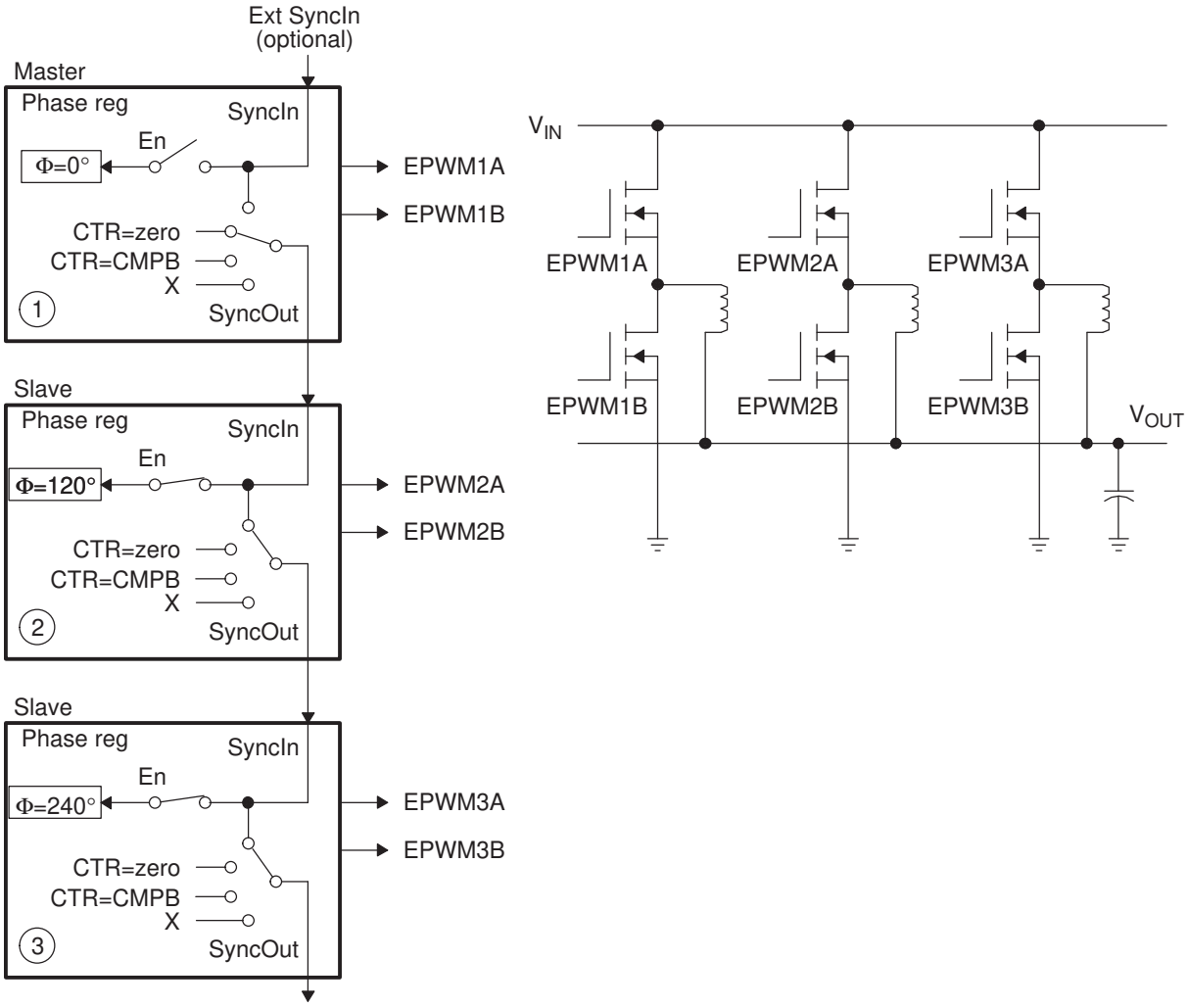
TBPHS(3,2) = (600/3) x (2-1) = 200 (i.e., Phase value for Slave module 2)

TBPHS(3,3) = 400 (i.e., Phase value for Slave module 3)

Figure 7-68 shows the waveforms for the configuration in Figure 7-67.

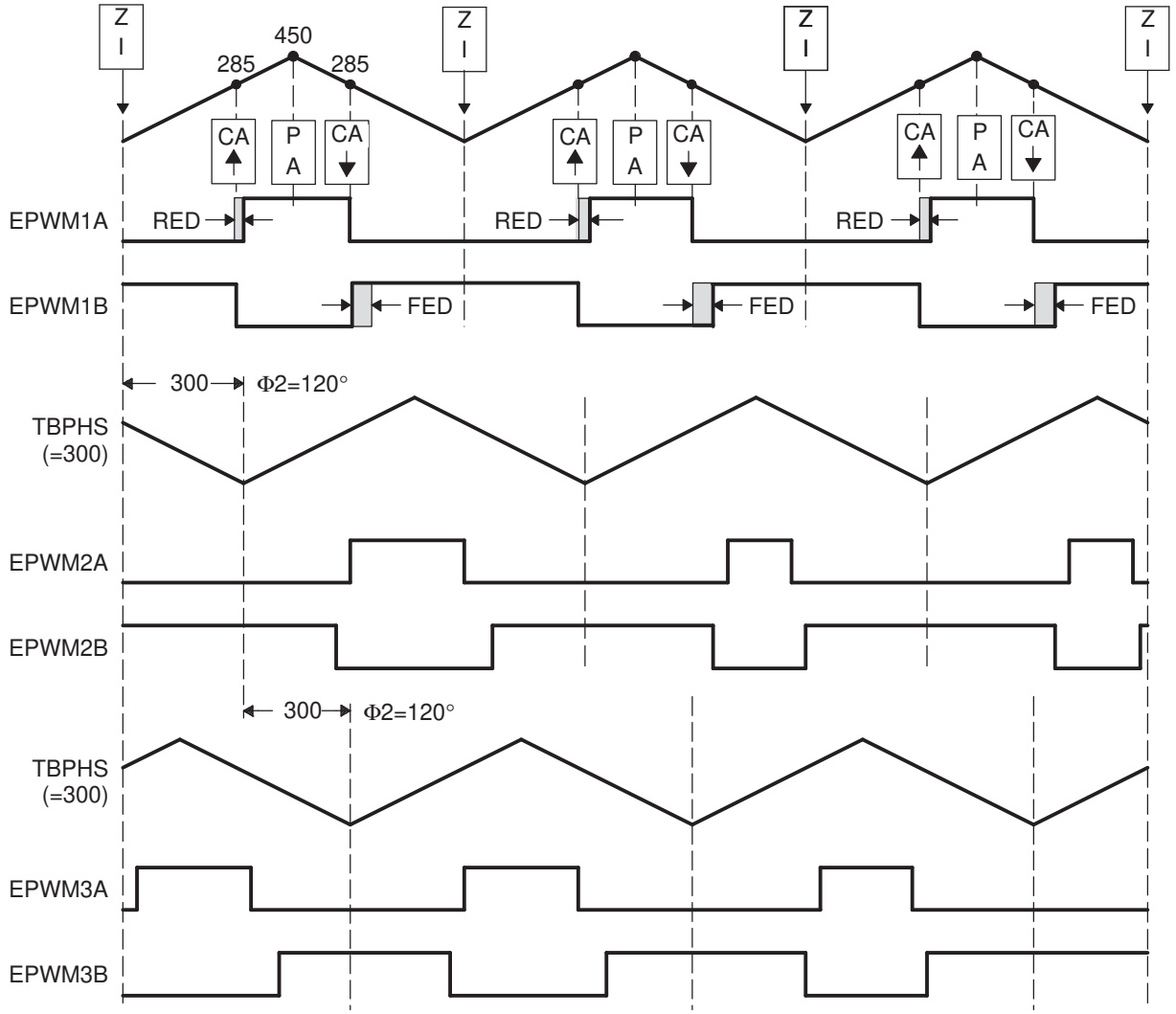
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-67. Control of a 3-Phase Interleaved DC/DC Converter



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-68. 3-Phase Interleaved DC/DC Converter Waveforms for Figure 7-67



Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-12. Code Snippet for Configuration in Figure 7-67

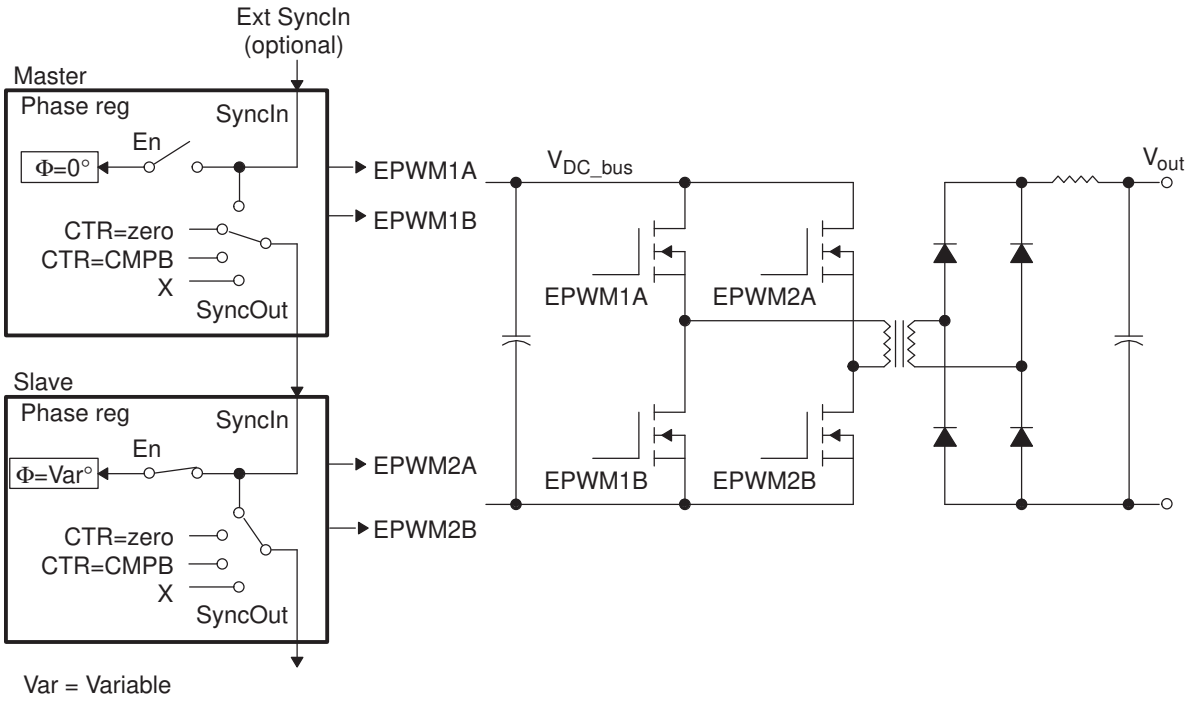
```
//=====================================================================
// Config
// Initialization Time
//=======================
// EPWM Module 1 config
EPwm1Regs.TBPRD = 450;                          // Period = 900 TBCLK counts
EPwm1Regs.TBPHS.half.TBPHS = 0; // Set Phase register to zero
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;  // Symmetrical mode
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Master module
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;     // Sync down-stream module
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;              // set actions for EPWM1A
EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm1Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;  // enable Dead-band module
EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;       // Active Hi complementary
EPwm1Regs.DBFED = 20;                           // FED = 20 TBCLKs
EPwm1Regs.DBRED = 20;                           // RED = 20 TBCLKs
// EPWM Module 2 config
EPwm2Regs.TBPRD = 450;                          // Period = 900 TBCLK counts
EPwm2Regs.TBPHS.half.TBPHS = 300;               // Phase = 300/900 * 360 = 120 deg
EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;  // Symmetrical mode
EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;          // Slave module
EPwm2Regs.TBCTL.bit.PHSDIR = TB_DOWN;           // Count DOWN on sync (=120 deg)
EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;      // sync flow-through
EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;              // set actions for EPWM2A
EPwm2Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm2Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;  // enable Dead-band module
EPwm2Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;       // Active Hi Complementary
EPwm2Regs.DBFED = 20;                           // FED = 20 TBCLKs
EPwm2Regs.DBRED = 20;                           // RED = 20 TBCLKs
// EPWM Module 3 config
EPwm3Regs.TBPRD = 450;                          // Period = 900 TBCLK counts
EPwm3Regs.TBPHS.half.TBPHS = 300;               // Phase = 300/900 * 360 = 120 deg
EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;  // Symmetrical mode
EPwm3Regs.TBCTL.bit.PHSEN = TB_ENABLE;          // Slave module
EPwm2Regs.TBCTL.bit.PHSDIR = TB_UP;             // Count UP on sync (=240 deg)
EPwm3Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;      // sync flow-through
EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm3Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm3Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm3Regs.AQCTLA.bit.CAU = AQ_SET;              // set actions for EPWM3Ai
EPwm3Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm3Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;  // enable Dead-band module
EPwm3Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;       // Active Hi complementary
EPwm3Regs.DBFED = 20;                           // FED = 20 TBCLKs
EPwm3Regs.DBRED = 20;                           // RED = 20 TBCLKs
// Run Time (Note: Example execution of one run-time instant)
//=========================================================
EPwm1Regs.CMPA.half.CMPA = 285;                 // adjust duty for output EPWM1A
EPwm2Regs.CMPA.half.CMPA = 285;                 // adjust duty for output EPWM2A
EPwm3Regs.CMPA.half.CMPA = 285;                 // adjust duty for output EPWM3A
```


Copyright © 2012–2019, Texas Instruments Incorporated

7.3.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter

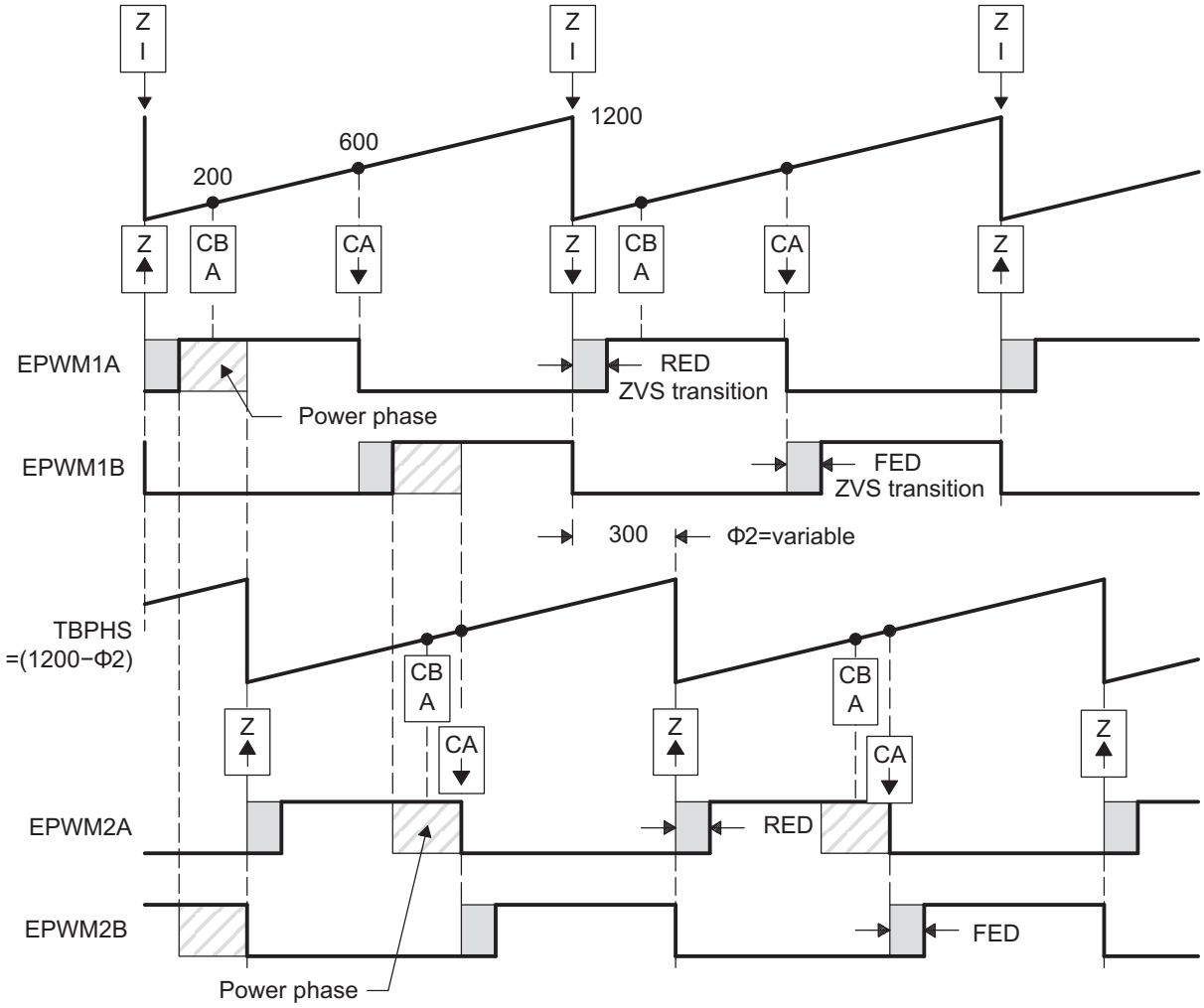
The example given in Figure 7-69 assumes a static or constant phase relationship between legs (modules). In such a case, control is achieved by modulating the duty cycle. It is also possible to dynamically change the phase value on a cycle-by-cycle basis. This feature lends itself to controlling a class of power topologies known as *phase-shifted full bridge*, or *zero voltage switched full bridge.* Here the controlled parameter is not duty cycle (this is kept constant at approximately 50 percent); instead it is the phase relationship between legs. Such a system can be implemented by allocating the resources of two PWM modules to control a single power stage, which in turn requires control of four switching elements. Figure 7-70 shows a master/slave module combination synchronized together to control a full H-bridge. In this case, both master and slave modules are required to switch at the same PWM frequency. The phase is controlled by using the slave's phase register (TBPHS). The master's phase register is not used and therefore can be initialized to zero.

Figure 7-69. Controlling a Full-H Bridge Stage (F_{PWM2} = $F_{PWM1)}$



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-70. ZVS Full-H Bridge Waveforms



Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-13. Code Snippet for Configuration in Figure 7-69

```
//=============================================================================
// Config
//=============================================================================
// Initialization Time
//=======================
// EPWM Module 1 config
EPwm1Regs.TBPRD = 1200;                                 // Period = 1201 TBCLK counts
EPwm1Regs.CMPA = 600;                                   // Set 50% fixed duty for EPWM1A
EPwm1Regs.TBPHS.half.TBPHS = 0;                         // Set Phase register to zero
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;              // Asymmetrical mode
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;                 // Master module
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;             // Sync down-stream module
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;           // load on CTR=Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;           // load on CTR=Zero
EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;                      // set actions for EPWM1A
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;          // enable Dead-band module
EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;               // Active Hi complementary
EPwm1Regs.DBFED = 50;                                   // FED = 50 TBCLKs initially
EPwm1Regs.DBRED = 70;                                   // RED = 70 TBCLKs initially
// EPWM Module 2 config
EPwm2Regs.TBPRD = 1200;                                 // Period = 1201 TBCLK counts
EPwm2Regs.CMPA.half.CMPA = 600;                         // Set 50% fixed duty EPWM2A
EPwm2Regs.TBPHS.half.TBPHS = 0;                         // Set Phase register to zero initially
EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;              // Asymmetrical mode
EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;                  // Slave module
EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;              // sync flow-through
EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;           // load on CTR=Zero
EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;           // load on CTR=Zero
EPwm2Regs.AQCTLA.bit.ZRO = AQ_SET;                      // set actions for EPWM2A
EPwm2Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm2Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;          // enable Dead-band module
EPwm2Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;               // Active Hi complementary
EPwm2Regs.DBFED = 30;                                   // FED = 30 TBCLKs initially
EPwm2Regs.DBRED = 40;                                   // RED = 40 TBCLKs initially
// Run Time (Note: Example execution of one run-time instant)
//=============================================================
EPwm2Regs.TBPHS = 1200-300;                             // Set Phase reg to 300/1200 * 360 = 90 deg
EPwm1Regs.DBFED = FED1_NewValue;                        // Update ZVS transition interval
EPwm1Regs.DBRED = RED1_NewValue;                        // Update ZVS transition interval
EPwm2Regs.DBFED = FED2_NewValue;                        // Update ZVS transition interval
EPwm2Regs.DBRED = RED2_NewValue;                        // Update ZVS transition interval
EPwm1Regs.CMPB = 200;                                   // adjust point-in-time for ADCSOC trigger
```

7.3.10 Controlling a Peak Current Mode Controlled Buck Module

Peak current control techniques offer a number of benefits like automatic over current limiting, fast correction for input voltage variations and reducing magnetic saturation. Figure 7-71 shows the use of ePWM1A along with the on-chip analog comparator for buck converter topology. The output current is sensed through a current sense resistor and fed to the positive terminal of the on-chip comparator. The internal programmable 10-bit DAC can be used to provide a reference peak current at the negative

Copyright © 2012–2019, Texas Instruments Incorporated

terminal of the comparator. Alternatively, an external reference could be connected at this input. The comparator output is an input to the Digital compare sub-module. The ePWM module is configured in such a way so as to trip the ePWM1A output as soon as the sensed current reaches the peak reference value. A cycle-by-cycle trip mechanism is used. Figure 7-72 shows the waveforms generated by the configuration.

Figure 7-71. Peak Current Mode Control of a Buck Converter

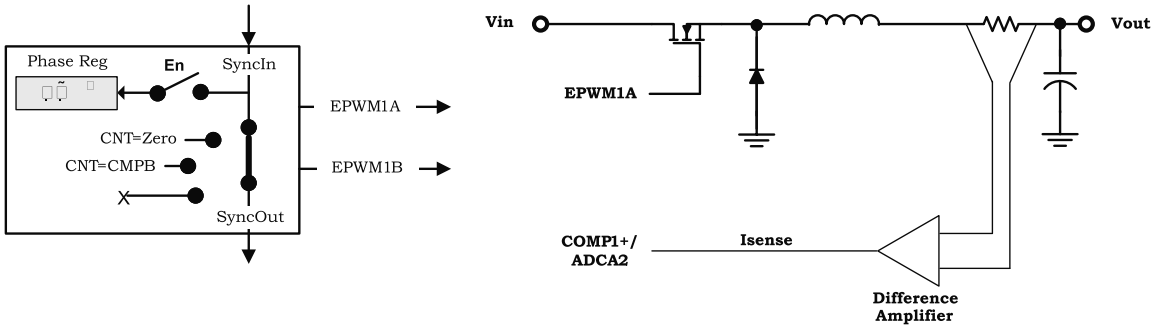
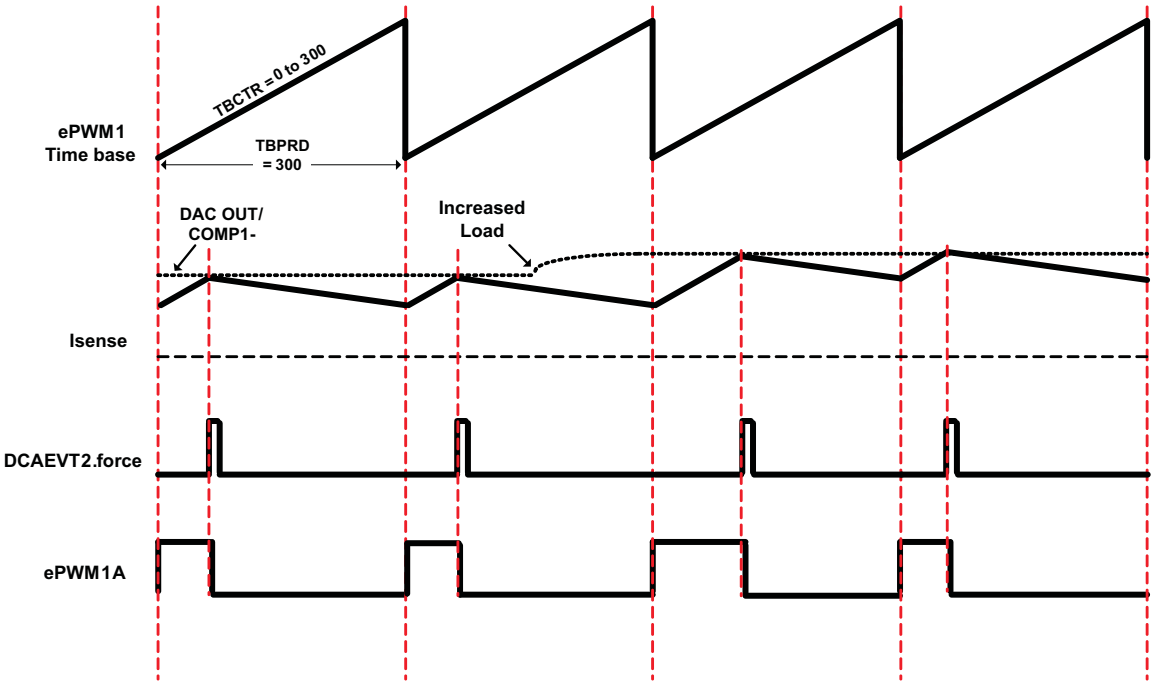


Figure 7-72. Peak Current Mode Control Waveforms for Figure 7-71



Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-14. Code Snippet for Configuration in Figure 7-71

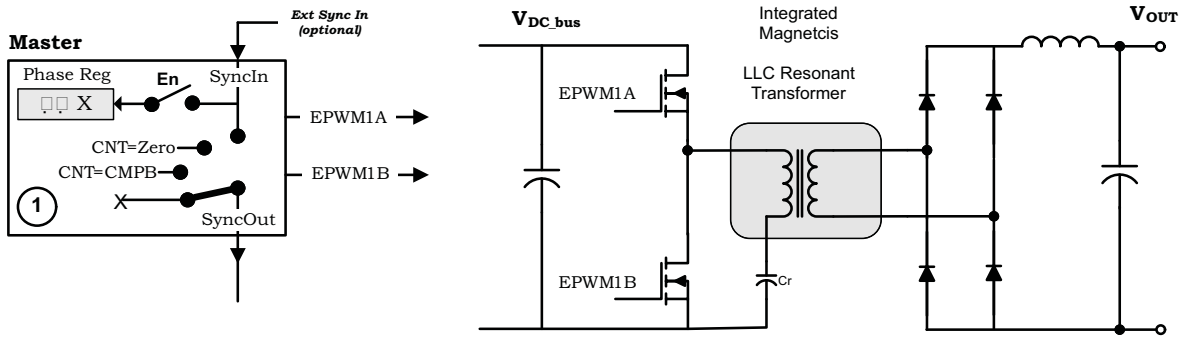
```
//=============================================================================
// Config //
// Initialization Time
//=============================================================================
EPwm1Regs.TBPRD = 300;
// Period = 300 TBCLK counts                                    // (200 KHz @ 60MHz clock)
EPwm1Regs.TBPHS.half.TBPHS = 0;                                 // Set Phase register to zero
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;                      // Asymmetrical mode
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;                         // Phase loading disabled
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;                        // Clock ratio to SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
Pwm1Regs.AQCTLA.bit.ZRO = AQ_SET;                               // Set PWM1A on Zero
                                                                // Define an event (DCAEVT2) based on
Comparator 1 Output EPwm1Regs.DCTRIPSEL.bit.DCAHCOMPSEL = DC_COMP1OUT;           // DCAH = Comparator
1 output
EPwm1Regs.TZDCSEL.bit.DCAEVT2 = TZ_DCAH_HI;                     // DCAEVT2 = DCAH high(will become
active
                                                                // as Comparator output goes high)
EPwm1Regs.DCACTL.bit.EVT2SRCSEL = DC_EVT2;                      // DCAEVT2 = DCAEVT2 (not filtered)
EPwm1Regs.DCACTL.bit.EVT2FRCSYNCSEL = DC_EVT_ASYNC;             // Take async path // Enable DCAEVT2 as
a one shot trip source
                                                                // Note: DCxEVT1 events can be defined
as one-shot.
                                                                // DCxEVT2 events can be defined as
cycle-by-
cycle. EPwm1Regs.TZSEL.bit.DCAEVT2 = 1;                               // What do we want the DCAEVT1
and DCBEVT1 events to do?
                                                                // DCAEVTx events can force EPWMxA //
DCBEVTx events can force EPWMxB
EPwm1Regs.TZCTL.bit.TZA = TZ_FORCE_LO;                          // EPWM1A will go low
//=============================================================================
// Run Time
//============================================================================= //
Adjust reference peak current to Comparator 1 negative input
```

7.3.11 Controlling H-Bridge LLC Resonant Converter

Various topologies of resonant converters are well-known in the field of power electronics for many years. In addition to these, H-bridge LLC resonant converter topology has recently gained popularity in many consumer electronics applications where high efficiency and power density are required. In this example single channel configuration of ePWM1 is detailed, yet the configuration can easily be extended to multi channel. Here the controlled parameter is not duty cycle (this is kept constant at approximately 50 percent); instead it is frequency. Although the deadband is not controlled and kept constant as 300ns (i.e 30 @100MHz TBCLK), it is up to user to update it in real time to enhance the efficiency by adjusting enough time delay for soft switching.

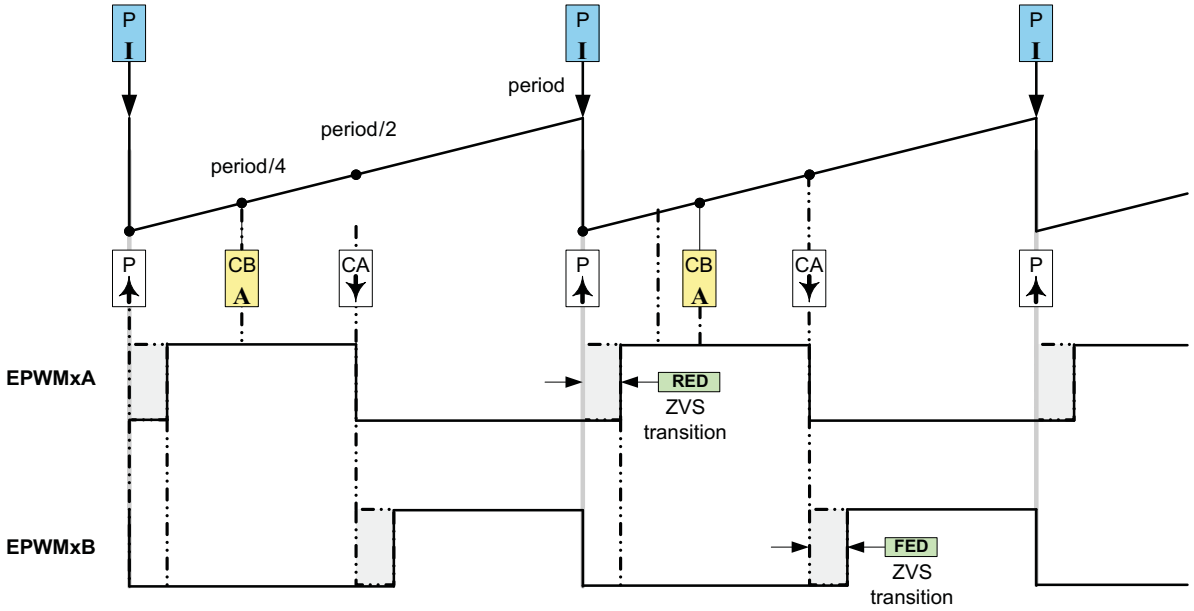
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-73. Control of Two Resonant Converter Stages



NOTE: Θ = X indicates value in phase register is a "don't care"

Figure 7-74. H-Bridge LLC Resonant Converter PWM Waveforms



P I Indicates this event triggers an interrupt

CB A Indicates this event triggers an ADC start of conversion

Copyright © 2012–2019, Texas Instruments Incorporated

Example 7-15. Code Snippet for Configuration in Figure 7-73

```
//=====================================================================
// Config
//===================================================================== //
Initialization Time
//======================== //
EPWMxA & EPWMxB config
EPwm1Regs.TBCTL.bit.PRDLD = TB_IMMEDIATE;                          // Set immediate load
EPwm1Regs.TBPRD = period;                                          // PWM frequency = 1 / period
EPwm1Regs.CMPA.half.CMPA = period/2;                               // Set duty as 50%
EPwm1Regs.CMPB = period/4;                                         // Set duty as 25%
EPwm1Regs.TBPHS.half.TBPHS = 0;                                    // Set as master, phase =0
EPwm1Regs.TBCTR = 0;                                               // Time base counter =0
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;                         // Count-
up mode: used for asymmetric PWM
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;                            // Disable phase loading
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;                        // Used to sync EPWM(n+1) "down-
stream"
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;                           // Set the clock rate
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;                              // Set the clock rate
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_PRD;                       // Load on CTR=PRD
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_PRD;                       // Load on CTR=PRD
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;                        // Shadow mode. Operates as a
double buffer.
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;                        // Shadow mode. Operates as a
double buffer.
EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;                                 // Set PWM1A on Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;                               // Clear PWM1A on event A, up
count
EPwm1Regs.AQCTLB.bit.CAU = AQ_SET;                                 // Set PWM1B on event A, up count
EPwm1Regs.AQCTLB.bit.PRD = AQ_CLEAR;                               // Clear PWM1B on PRD
EPwm1Regs.DBCTL.bit.IN_MODE = DBA_ALL;                             // EPWMxA is the source for both
delays
EPwm1Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;                     // Enable Dead-band module
EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;                          // Active High Complementary (AHC)
EPwm1Regs.DBRED = 30;                                              // RED = 30 TBCLKs initially
EPwm1Regs.DBFED = 30;                                              // FED = 30 TBCLKs initially
                                                                   // Configure TZ1 for short cct

protection EALLOW;
EPwm1Regs.TZSEL.bit.OSHT1 = 1;                                     // one-
shot source EPwm1Regs.TZCTL.bit.TZA = TZ_FORCE_LO;
                                                                   // set EPWM1A to low at fault
EPwm1Regs.TZCTL.bit.TZB = TZ_FORCE_LO;                             // set EPWM1B to low at fault
instant
EPwm1Regs.TZEINT.bit.OST = 1;                                      // Enable TZ interrupt EDIS;
                                                                   // Enable HiRes option EALLOW;

EPwm1Regs.HRCNFG.all = 0x0;
EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;
EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;
EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_PRD; EDIS;                    // Run Time (Note: Example
execution of one run-time instant)
//=============================================================
EPwm1Regs.TBPRD = period_new value;                                // Update new period
EPwm1Regs.CMPA.half.CMPA= period_new value/2;
                                                                   // Update new CMPA EPwm1Regs.CMPB=

period_new value/4;
                                                                   // Update new CMPB
                                                                  // Update new CMPB
```


Copyright © 2012–2019, Texas Instruments Incorporated

7.4 Registers

This chapter includes the register layouts and bit description for the submodules.

7.4.1 Time-Base Submodule Registers

Figure 7-75 through Figure 7-86 and Table 7-25 through Table 7-36 provide the time-base register definitions.

Figure 7-75. Time-Base Period and Mirror 2 Register (TBPRD / TBPRDM2)

15	0
TBPRD	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-25. Time-Base Period and Mirror 2 Register (TBPRD / TBPRDM2) Field Descriptions

Bit	Field	Value	Description
15-0	TBPRD	0000-FFFFh	These bits determine the period of the time-base counter. This sets the PWM frequency. Shadowing of this register is enabled and disabled by the TBCTL[PRDLD] bit. By default this register is shadowed. • If TBCTL[PRDLD] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the active register will be loaded from the shadow register when the time-base counter equals zero. • If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware. • The active and shadow registers share the same memory map address.

Figure 7-76. Time-Base Period High-Resolution and Mirror 2 Register (TBPRDHR / TBPRDHRM2)

15	8
TBPRDHR	
R/W-0	
7	0
Reserved	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-26. Time-Base Period High-Resolution and Mirror 2 Register (TBPRDHR / TBPRDHRM2) Field Descriptions

Bit	Field	Value	Description
15-8	TBPRDHR	00-FFh	Period High Resolution Bits These 8-bits contain the high-resolution portion of the period value. The TBPRDHR register is not affected by the TBCTL[PRDLD] bit. Reads from this register always reflect the shadow register. Likewise writes are also to the shadow register. The TBPRDHR register is only used when the high resolution period feature is enabled. This register is only available with ePWM modules which support high-resolution period control.
7-0	Reserved	0	Reserved

Figure 7-77. Time-Base Period Mirror Register (TBPRDM)

15	0
TBPRD	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-27. Time-Base Period Mirror Register (TBPRDM) Field Descriptions

Bit	Field	Value	Description
15-0	TBPRD	0000-FFFFh	TBPRDM and TBPRD can both be used to access the time-base period. TBPRD provides backwards compatibility with earlier ePWM modules. The mirror registers (TBPRDM and TBPRDHRM) allow for 32-bit writes to TBPRDHR in one access. Due to the odd address memory location of the TBPRD legacy register, a 32-bit write is not possible. By default writes to this register are shadowed. Unlike the TBPRD register, reads of TBPRDM always return the active register value. Shadowing is enabled and disabled by the TBCTL[PRDLD] bit. • If TBCTL[PRDLD] = 0, then the shadow is enabled and any write will automatically go to the shadow register. In this case the active register will be loaded from the shadow register when the time-base counter equals zero. Reads return the active value. • If TBCTL[PRDLD] = 1, then the shadow is disabled and any write to this register will go directly to the active register controlling the hardware. Likewise reads return the active value.

Figure 7-78. Time-Base Period High-Resolution Mirror Register (TBPRDHRM)

15	8
TBPRDHR	
R/W-0	

7	0
Reserved	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-28. Time-Base Period High-Resolution Mirror Register (TBPRDHRM) Field Descriptions

Bit	Field	Value	Description
15-8	TBPRDHR	00-FFh	Period High Resolution Bits These 8-bits contain the high-resolution portion of the period value TBPRD provides backwards compatibility with earlier ePWM modules. The mirror registers (TBPRDM and TBPRDHRM) allow for 32-bit writes to TBPRDHR in one access. Due to the odd-numbered memory address location of the TBPRD legacy register, a 32-bit write is not possible with TBPRD and TBPRDHR. The TBPRDHRM register is not affected by the TBCTL[PRDLD] bit Writes to both the TBPRDHR and TBPRDM locations access the high-resolution (least significant 8-bit) portion of the Time Base Period value. The only difference is that unlike TBPRDHR, reads from the mirror register TBPRDHRM, are indeterminate (reserved for TI Test). The TBPRDHRM register is available with ePWM modules which support high-resolution period control and is used only when the high resolution period feature is enabled.
7-0	Reserved	00-FFh	Reserved for TI Test

Figure 7-79. Time-Base Phase Register and Mirror Register (TBPHS / TBPHSM)

15	0
TBPHS	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-29. Time-Base Phase Register and Mirror Register (TBPHS / TBPHSM) Field Descriptions

Bit	Field	Value	Description
15-0	TBPHS	0000-FFFF	These bits set time-base counter phase of the selected ePWM relative to the time-base that is supplying the synchronization input signal. • If TBCTL[PHSEN] = 0, then the synchronization event is ignored and the time-base counter is not loaded with the phase. • If TBCTL[PHSEN] = 1, then the time-base counter (TBCTR) will be loaded with the phase (TBPHS) when a synchronization event occurs. The synchronization event can be initiated by the input synchronization signal (EPWMxSYNCI) or by a software forced synchronization.

Figure 7-80. Time-Base Phase High-Resolution Register and Mirror Register (TBPHSHR / TBPHSHRM)

15	8
TBPHSHR	
R/W-0	

7	0
Reserved	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-30. Time-Base Phase High-Resolution Register and Mirror Register (TBPHSHR / TBPHSHRM) Field Descriptions

Bit	Field	Value	Description
15-8	TBPHSHR	00-FFh	Time base phase high-resolution bits
7-0	Reserved		Reserved

Figure 7-81. Time-Base Counter Register (TBCTR)

15	0
TBCTR	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-31. Time-Base Counter Register (TBCTR) Field Descriptions

Bit	Field	Value	Description
15-0	TBCTR	0000-FFFF	Reading these bits gives the current time-base counter value. Writing to these bits sets the current time-base counter value. The update happens as soon as the write occurs; the write is NOT synchronized to the time-base clock (TBCLK) and the register is not shadowed.

Figure 7-82. Time-Base Control Register (TBCTL)

15	14	13	12	10	9	8
FREE, SOFT		PHSDIR	CLKDIV		HSPCLKDIV	
R/W-0		R/W-0	R/W-0		R/W-0,0,1	

7	6	5	4	3	2	1	0
HSPCLKDIV	SWFSYNC	SYNCOSEL		PRDLD	PHSEN	CTRMODE	
R/W-0,0,1	R/W-0	R/W-0		R/W-0	R/W-0	R/W-11	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-32. Time-Base Control Register (TBCTL) Field Descriptions

Bit	Field	Value	Description
15-14	FREE, SOFT		Emulation Mode Bits. These bits select the behavior of the ePWM time-base counter during emulation events:
		00	Stop after the next time-base counter increment or decrement
		01	Stop when counter completes a whole cycle: • Up-count mode: stop when the time-base counter = period (TBCTR = TBPRD) • Down-count mode: stop when the time-base counter = 0x00 (TBCTR = 0x00) • Up-down-count mode: stop when the time-base counter = 0x00 (TBCTR = 0x00)
		1X	Free run
13	PHSDIR		Phase Direction Bit. This bit is only used when the time-base counter is configured in the up-down-count mode. The PHSDIR bit indicates the direction the time-base counter (TBCTR) will count after a synchronization event occurs and a new phase value is loaded from the phase (TBPHS) register. This is irrespective of the direction of the counter before the synchronization event.. In the up-count and down-count modes this bit is ignored.
		0	Count down after the synchronization event.
		1	Count up after the synchronization event.
12-10	CLKDIV		Time-base Clock Prescale Bits These bits determine part of the time-base clock prescale value. TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV)
		000	/1 (default on reset)
		001	/2
		010	/4
		011	/8
		100	/16
		101	/32
		110	/64
		111	/128
9-7	HSPCLKDIV		High Speed Time-base Clock Prescale Bits These bits determine part of the time-base clock prescale value. TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV) This divisor emulates the HSPCLK in the TMS320x281x system as used on the Event Manager (EV) peripheral.
		000	/1
		001	/2 (default on reset)
		010	/4
		011	/6
		100	/8
		101	/10
		110	/12
		111	/14
6	SWFSYNC		Software Forced Synchronization Pulse
		0	Writing a 0 has no effect and reads always return a 0.
		1	Writing a 1 forces a one-time synchronization pulse to be generated. This event is ORed with the EPWMxSYNCI input of the ePWM module. SWFSYNC is valid (operates) only when EPWMxSYNCI is selected by SYNCOSEL = 00.
5-4	SYNCOSEL		Synchronization Output Select. These bits select the source of the EPWMxSYNCO signal.
		00	EPWMxSYNC:
		01	CTR = zero: Time-base counter equal to zero (TBCTR = 0x00)
		10	CTR = CMPB : Time-base counter equal to counter-compare B (TBCTR = CMPB)
		11	Disable EPWMxSYNCO signal

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-32. Time-Base Control Register (TBCTL) Field Descriptions (continued)

Bit	Field	Value	Description
3	PRDLD		Active Period Register Load From Shadow Register Select
		0	The period register (TBPRD) is loaded from its shadow register when the time-base counter, TBCTR, is equal to zero. A write or read to the TBPRD register accesses the shadow register.
		1	The period register (TBPRD) is loaded from its shadow register when the time-base counter TBCTR, is equal to zero and/or a sync event as determined by the TBCTL2[PRDLDSYNC] bit. A write/read to the TBPRD register accesses the shadow register.
2	PHSEN		Counter Register Load From Phase Register Enable
		0	Do not load the time-base counter (TBCTR) from the time-base phase register (TBPHS).
		1	Allow Counter to be loaded from the Phase register (TBPHS) and shadow to active load events when an EPWMxSYNCI input signal occurs or a software-forced sync signal, see bit 6.
1-0	CTRMODE		Counter Mode The time-base counter mode is normally configured once and not changed during normal operation. If you change the mode of the counter, the change will take effect at the next TBCLK edge and the current counter value shall increment or decrement from the value before the mode change. These bits set the time-base counter mode of operation as follows:
		00	Up-count mode
		01	Down-count mode
		10	Up-down-count mode
		11	Stop-freeze counter operation (default on reset)

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-83. Time-Base Status Register (TBSTS)

15				8
Reserved				
R-0				

7	3	2	1	0
Reserved		CTRMAX	SYNCI	CTRDIR
R-0		R/W1C-0	R/W1C-0	R-1

LEGEND: R/W = Read/Write; R = Read only; R/W1C = Read/Write 1 to clear; -*n* = value after reset

Table 7-33. Time-Base Status Register (TBSTS) Field Descriptions

Bit	Field	Value	Description
15:3	Reserved		Reserved
2	CTRMAX		Time-Base Counter Max Latched Status Bit
		0	Reading a 0 indicates the time-base counter never reached its maximum value. Writing a 0 will have no effect.
		1	Reading a 1 on this bit indicates that the time-base counter reached the max value 0xFFFF. Writing a 1 to this bit will clear the latched event.
1	SYNCI		Input Synchronization Latched Status Bit
		0	Writing a 0 will have no effect. Reading a 0 indicates no external synchronization event has occurred.
		1	Reading a 1 on this bit indicates that an external synchronization event has occurred (EPWMxSYNCI). Writing a 1 to this bit will clear the latched event.
0	CTRDIR		Time-Base Counter Direction Status Bit. At reset, the counter is frozen; therefore, this bit has no meaning. To make this bit meaningful, you must first set the appropriate mode via TBCTL[CTRMODE].
		0	Time-Base Counter is currently counting down.
		1	Time-Base Counter is currently counting up.

Figure 7-84. High-Resolution Period Control Register (HRPCTL)

15				8
Reserved				
R-0				

7	3	2	1	0
Reserved		TBPHSHR LOADE	Reserved	HRPE
R-0		R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-34. High-Resolution Period Control Register (HRPCTL) Field Descriptions

Bit	Field	Value	Description[(1) (2)]
15-3	Reserved		Reserved

(1) This register is EALLOW protected.
(2) This register is used with Type 1 ePWM modules (support high-resolution period) only.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-34. High-Resolution Period Control Register (HRPCTL) Field Descriptions (continued)

Bit	Field	Value	Description[1] [2]
2	TBPHSHRLOADE		TBPHSHR Load Enable
			This bit allows you to synchronize ePWM modules with a high-resolution phase on a SYNCIN, TBCTL[SWFSYNC], or digital compare event. This allows for multiple ePWM modules operating at the same frequency to be phase aligned with high-resolution.
		0	Disables synchronization of high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital compare event.
		1	Synchronize the high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital comparator synchronization event. The phase is synchronized using the contents of the high-resolution phase TBPHSHR register.
			The TBCTL[PHSEN] bit which enables the loading of the TBCTR register with TBPHS register value on a SYNCIN, or TBCTL[SWFSYNC] event works independently. However, users need to enable this bit also if they want to control phase in conjunction with the high-resolution period feature.
			Note: This bit and the TBCTL[PHSEN] bit must be set to 1 when high resolution period control is enabled for up-down count mode even if TBPHSHR = 0x00.
1	Reserved		Reserved
0	HRPE		High Resolution Period Enable Bit
		0	High resolution period feature disabled. In this mode the ePWM behaves as a Type 0 ePWM.
		1	High resolution period enabled. In this mode the HRPWM module can control high-resolution of both the duty and frequency.
			When high-resolution period is enabled, TBCTL[CTRMODE] = 0,1 (down-count mode) is not supported.

Figure 7-85. Time-Base Control Register 2 (TBCTL2)

15 14	13 0
PRDLDSYNC	Reserved
R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-35. Time-Base Control Register 2 (TBCTL2) Field Descriptions

Bit	Field	Value	Description
15-14	PRDLDSYNC		Shadow to Active Period Register Load on SYNC Event
		00	Shadow to Active Load of TBRD occurs only when TBCTR = 0 (same as legacy).
		01	Shadow to Active Load of TBRD occurs both when TBCTR = 0 and when SYNC occurs.
		10	Shadow to Active Load of TBPRD occurs only when a SYNC is received.
		11	Reserved
			Note: This bit selection is valid only if TBCTL[PRDLD]=0.
13-0	Reserved		Reserved

Figure 7-86. EPWMx Link Register (EPWMXLINK)

31 20	19 16
Reserved	CMPDLINK
	R/W-0

15 12	11 8	7 4	3 0
CMPCLINK	CMPBLINK	CMPALINK	TBPRDLINK
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-36. EPWMx Link Register (EPWMXLINK) Field Descriptions

Bit	Field	Value	Description
30-20	Reserved		Reserved
19-16	CMPDLINK		CMPD Link Bits
			Writes to the CMPD register in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's CMPD register.
		0010	ePWM1
		0011	ePWM2
		0100	ePWM3
		0101	ePWM4
		0110	ePWM5
		0111	ePWM6
		1000	ePWM7
		1001	ePWM8
		1010	ePWM9
			All other values are reserved and do nothing (i.e., register is linked to itself).
15-12	CMPCLINK		CMPC Link Bits
			Writes to the CMPC register in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's CMPC register.
		0010	ePWM1
		0011	ePWM2
		0100	ePWM3
		0101	ePWM4
		0110	ePWM5
		0111	ePWM6
		1000	ePWM7
		1001	ePWM8
		1010	ePWM9
			All other values are reserved and do nothing (i.e., register is linked to itself).
11-8	CMPBLINK		CMPB:CMPBHR Link Bits
			Writes to the CMPB:CMPBHR registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's CMPB:CMPBHR registers.
		0010	ePWM1
		0011	ePWM2
		0100	ePWM3
		0101	ePWM4
		0110	ePWM5
		0111	ePWM6
		1000	ePWM7
		1001	ePWM8
		1010	ePWM9
			All other values are reserved and do nothing (i.e., register is linked to itself).

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-36. EPWMx Link Register (EPWMXLINK) Field Descriptions (continued)

Bit	Field	Value	Description
7-4	CMPALINK		CMPA:CMPAHR Link Bits Writes to the CMPA:CMPAHR registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's CMPA:CMPAHR registers.
		0010	ePWM1
		0011	ePWM2
		0100	ePWM3
		0101	ePWM4
		0110	ePWM5
		0111	ePWM6
		1000	ePWM7
		1001	ePWM8
		1010	ePWM9
			All other values are reserved and do nothing (i.e., register is linked to itself).
3-0	TBPRDLINK		TBPRD:TBPRDHR Link Bits Writes to the TBPRD:TBPRDHR registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's TBPRD:TBPRDHR registers.
		0010	ePWM1
		0011	ePWM2
		0100	ePWM3
		0101	ePWM4
		0110	ePWM5
		0111	ePWM6
		1000	ePWM7
		1001	ePWM8
		1010	ePWM9
			All other values are reserved and do nothing (i.e., register is linked to itself).

Copyright © 2012–2019, Texas Instruments Incorporated

7.4.2 Counter-Compare Submodule Registers

Figure 7-87 through Figure 7-97 and Table 7-37 through Table 7-48 illustrate the counter-compare submodule control and status registers.

Figure 7-87. Counter-Compare Control Register (CMPCTL)

15	14	13	12	11	10	9	8
Reserved		LOADBSYNC		LOADASYNC		SHDWBFULL	SHDWAFULL
R-0							
7	6	5	4	3	2	1	0
Reserved	SHDWBMODE	Reserved	SHDWAMODE	LOADBMODE		LOADAMODE	
R-0		R-0					

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-37. Counter-Compare Control Register (CMPCTL) Field Descriptions

Bit	Field	Value	Description
15-14	Reserved	0	Reserved
13-12	LOADBSYNC		Shadow to Active CMPB Register Load on SYNC event
		00	Shadow to Active Load of CMPB:CMPBHR occurs according to LOADBMODE (bits 1,0) (same as legacy)
		01	Shadow to Active Load of CMPB:CMPBHR occurs both according to LOADBMODE bits and when SYNC occurs
		10	Shadow to Active Load of CMPB:CMPBHR occurs only when a SYNC is received.
		11	Reserved
			Note: This bit is valid only if CMPCTL[SHDWBMODE] = 0.
11-10	LOADASYNC		Shadow to Active CMPA Register Load on SYNC event
		00	Shadow to Active Load of CMPA:CMPAHR occurs according to LOADAMODE (bits 1,0) (same as legacy)
		01	Shadow to Active Load of CMPA:CMPAHR occurs both according to LOADAMODE bits and when SYNC occurs.
		10	Shadow to Active Load of CMPA:CMPAHR occurs only when a SYNC is received.
		11	Reserved
			Note: This bit is valid only if CMPCTL[SHDWAMODE] = 0.
9	SHDWBFULL		Counter-compare B (CMPB) Shadow Register Full Status Flag. This bit self clears once a load-strobe occurs.
		0	CMPB shadow FIFO not full yet
		1	Indicates the CMPB shadow FIFO is full; a CPU write will overwrite current shadow value.
8	SHDWAFULL		Counter-compare A (CMPA) Shadow Register Full Status Flag The flag bit is set when a 32-bit write to CMPA:CMPAHR register or a 16-bit write to CMPA register is made. A 16-bit write to CMPAHR register will not affect the flag.
			This bit self clears once a load-strobe occurs.
		0	CMPA shadow FIFO not full yet
		1	Indicates the CMPA shadow FIFO is full, a CPU write will overwrite the current shadow value.
7	Reserved	0	Reserved
6	SHDWBMODE		Counter-compare B (CMPB) Register Operating Mode
		0	Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.
		1	Immediate mode. Only the active compare B register is used. All writes and reads directly access the active register for immediate compare action
5	Reserved		Reserved
4	SHDWAMODE		Counter-compare A (CMPA) Register Operating Mode
		0	Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.
		1	Immediate mode. Only the active compare register is used. All writes and reads directly access the active register for immediate compare action

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-37. Counter-Compare Control Register (CMPCTL) Field Descriptions (continued)

Bit	Field	Value	Description
3-2	LOADBMODE		Active Counter-Compare B (CMPB) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWBMODE] = 1).
		00	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		01	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		10	Load on either CTR = Zero or CTR = PRD
		11	Freeze (no loads possible)
1-0	LOADBMODE		Active Counter-Compare A (CMPA) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1).
		00	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		01	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		10	Load on either CTR = Zero or CTR = PRD
		11	Freeze (no loads possible)

Figure 7-88. Compare Control Register (CMPCTL2)

15	14	13	12	11	10	9	8
Reserved		LOADDSYNC		LOADCSYNC		Reserved	
R-0		R/W		R/W		R-0	

7	6	5	4	3	2	1	0
Reserved	SHDWDMODE	Reserved	SHDWCMODE	LOADDMODE		LOADCMODE	
R-0	R/W	R-0	R/W	R/W		R/W	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 7-38. Counter-Compare Control Register (CMPCTL2) Field Descriptions

Bit	Field	Value	Description
15-14	Reserved	0	Reserved
13-12	LOADDSYNC		Shadow to Active CMPD Register Load on SYNC event
		00	Shadow to Active Load of CMPD occurs according to LOADDMODE
		01	Shadow to Active Load of CMPD occurs both according to LOADDMODE bits and when SYNC occurs.
		10	Shadow to Active Load of CMPD occurs only when a SYNC is received.
		11	Reserved
			Note: This bit is valid only if CMPCTL2[SHDWDMODE] = 0.
11-10	LOADCSYNC		Shadow to Active CMPC Register Load on SYNC event
		00	Shadow to Active Load of CMPC occurs according to LOADCMODE
		01	Shadow to Active Load of CMPC occurs both according to LOADCMODE bits and when SYNC occurs.
		10	Shadow to Active Load of CMPC occurs only when a SYNC is received.
		11	Reserved
			Note: This bit is valid only if CMPCTL2[SHDWCMODE] = 0.
9-7	Reserved		Reserved
6	SHDWDMODE		Compare D Register operating mode:
		0	Shadow mode – operates as a double buffer. All writes via the CPU access Shadow register.
		1	Immediate mode – only the Active compare register is used. All writes/reads via the CPU directly access the Active register for immediate “Compare action”.
5	Reserved	0	Reserved
4	SHDWCMODE		Counter-Compare C Register operating mode
		0	Shadow mode – operates as a double buffer. All writes via the CPU access Shadow register.
		0	Immediate mode – only the Active compare register is used. All writes/reads via the CPU directly access the Active register for immediate “Compare action”.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-38. Counter-Compare Control Register (CMPCTL2) Field Descriptions (continued)

Bit	Field	Value	Description
3-2	LOADDMODE		Active Counter-Compare D Load from Shadow Select Mode:
		00	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		01	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		10	Load on either CTR = Zero or CTR = PRD
		11	Freeze (no loads possible)
			Note: Has no effect in Immediate mode.
1-0	LOADCMODE		Active Compare C Load from Shadow Select Mode:
		00	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		01	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		10	Load on either CTR = Zero or CTR = PRD
		11	Freeze (no loads possible)
			Note: Has no effect in Immediate mode.

Figure 7-89. Compare A High-Resolution and Mirror 2 Register (CMPAHR / CMPAHRM2)

15							8
CMPAHR							
R/W-0							
7							0
Reserved							
R-0							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-39. Compare A High-Resolution and Mirror 2 Register (CMPAHR / CMPAHRM2) Field Descriptions

Bit	Field	Value	Description
15-8	CMPAHR	00-FFh	These 8-bits contain the high-resolution portion (least significant 8-bits) of the counter-compare A value. CMPA:CMPAHR can be accessed in a single 32-bit read/write. Shadowing is enabled and disabled by the CMPCTL[SHDWAMODE] bit as described for the CMPA register.
7-0	Reserved		Reserved for TI Test

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-90. Counter-Compare A and Mirror 2 Register (CMPA / CMPAM2)

15	0
CMPA	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-40. Counter-Compare A and Mirror 2 Register (CMPA / CMPAM2) Field Descriptions

Bit	Field	Description
15-0	CMPA	The value in the active CMPA register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare A" event. This event is sent to the action-qualifier where it is qualified and converted it into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include: • Do nothing; the event is ignored. • Clear: Pull the EPWMxA and/or EPWMxB signal low • Set: Pull the EPWMxA and/or EPWMxB signal high • Toggle the EPWMxA and/or EPWMxB signal Shadowing of this register is enabled and disabled by the CMPCTL[SHDWAMODE] bit. By default this register is shadowed. • If CMPCTL[SHDWAMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register. • Before a write, the CMPCTL[SHDWAFULL] bit can be read to determine if the shadow register is currently full. • If CMPCTL[SHDWAMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware. • In either mode, the active and shadow registers share the same memory map address.

Figure 7-91. Compare A High-Resolution Mirror Register (CMPAHRM)

15	8
CMPAHR	
R/W-0	

7	0
Reserved	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-41. Compare A High-Resolution Mirror Register (CMPAHRM) Field Descriptions

Bit	Field	Value	Description
15-8	CMPAHR	00-FFh	Compare A High-Resolution Bits Writes to both the CMPAHR and CMPAHRM locations access the high-resolution (least significant 8-bit) portion of the Counter Compare A value. The only difference is that unlike CMPAHR, reads from the mirror register, CMPAHRM, are indeterminate (reserved for TI Test). By default writes to this register are shadowed. Shadowing is enabled and disabled by the CMPCTL[SHDWAMODE] bit as described for the CMPAM register.
7-0	Reserved		Reserved for TI test.

Figure 7-92. Counter-Compare A Mirror Register (CMPAM)

15	0
CMPA	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-42. Counter-Compare A Mirror Register (CMPAM) Field Descriptions

Bit	Field	Value	Description
15-0	CMPA	0000-FFFFh	CMPA and CMPAM can both be used to access the counter-compare A value. The only difference is that the mirror register always reads back the active value. By default writes to this register are shadowed. Unlike the CMPA register, reads of CMPAM always return the active register value. Shadowing is enabled and disabled by the CMPCTL[SHDWAMODE] bit. • If CMPCTL[SHDWAMODE] = 0, then the shadow is enabled and any write will automatically go to the shadow register. All reads will reflect the active register value. In this case, the CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register. • Before a write, the CMPCTL[SHDWAFULL] bit can be read to determine if the shadow register is currently full. • If CMPCTL[SHDWAMODE] = 1, then the shadow register is disabled and any write will go directly to the active register, that is the register actively controlling the hardware.

Figure 7-93. Counter-Compare B Register (CMPBM)

15	0
CMPB	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 7-43. Counter-Compare B Register (CMPBM) Field Descriptions

Bit	Field	Value	Description
15-0	CMPB	0000-FFFFh	CMPB and CMPBM can both be used to access the counter-compare A value. The only difference is that the mirror register always reads back the active value. By default writes to this register are shadowed. Unlike the CMPB register, reads of CMPBM always return the active register value. Shadowing is enabled and disabled by the CMPCTL[SHDWBMODE] bit. • If CMPCTL[SHDWBMODE] = 0, then the shadow is enabled and any write will automatically go to the shadow register. All reads will reflect the active register value. In this case, the CMPCTL[LOADBMODE] bit field determines which event will load the active register from the shadow register. • Before a write, the CMPCTL[SHDWBFULL] bit can be read to determine if the shadow register is currently full. • If CMPCTL[SHDWBMODE] = 1, then the shadow register is disabled and any write will go directly to the active register, that is the register actively controlling the hardware.

Figure 7-94. Counter-Compare B Register (CMPB)

15	0
CMPB	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-44. Counter-Compare B Register (CMPB) Field Descriptions

Bit	Field	Description
15-0	CMPB	The value in the active CMPB register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare B" event. This event is sent to the action-qualifier where it is qualified and converted it into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include: • Do nothing. event is ignored. • Clear: Pull the EPWMxA and/or EPWMxB signal low • Set: Pull the EPWMxA and/or EPWMxB signal high • Toggle the EPWMxA and/or EPWMxB signal Shadowing of this register is enabled and disabled by the CMPCTL[SHDWBMODE] bit. By default this register is shadowed. • If CMPCTL[SHDWBMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADBMODE] bit field determines which event will load the active register from the shadow register: • Before a write, the CMPCTL[SHDWBFULL] bit can be read to determine if the shadow register is currently full. • If CMPCTL[SHDWBMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware. • In either mode, the active and shadow registers share the same memory map address.

Figure 7-95. Counter-Compare C Register (CMPC)

15	0
CMPC	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-45. Counter-Compare C Register (CMPC) Field Descriptions

Bit	Field	Description
15-0	CMPC	The value in the active CMPC register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare C" event. Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWCMODE] bit. By default this register is shadowed. • If CMPCTL2[SHDWCMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL2[LOADCMODE] bit field determines which event will load the active register from the shadow register: • If CMPCTL2[SHDWCMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register; that is, the register actively controlling the hardware. • In either mode, the active and shadow registers share the same memory map address.

Figure 7-96. Counter-Compare D Register (CMPD)

15	0
CMPD	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-46. Counter-Compare D Register (CMPD) Field Descriptions

Bit	Field	Description
15-0	CMPD	The value in the active CMPD register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare D" event. Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWDMODE] bit. By default this register is shadowed. • If CMPCTL2[SHDWDMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL2[LOADDMODE] bit field determines which event will load the active register from the shadow register: • If CMPCTL2[SHDWDMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware. • In either mode, the active and shadow registers share the same memory map address.

Figure 7-97. Compare B High-Resolution Register (CMPBHR)

15	8
CMPBHR	
R/W-0	

7	0
Reserved	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-47. Compare B High-Resolution Register (CMPBHR) Field Descriptions

Bit	Field	Value	Description
15-8	CMPBHR	00-FFh	Compare B High-Resolution Bits These 8-bits contain the high-resolution portion (least significant 8-bits) of the counter-compare B value. CMPB:CMPBHR can be accessed in a single 32-bit read/write. Shadowing is enabled and disabled by the CMPCTL[SHDWAMODE] bit as described for the CMPA register.
7-0	Reserved		Reserved for TI test.

Figure 7-98. Compare B High-Resolution Mirror Register (CMPBHRM)

15	8
CMPBHR	
R/W-0	

7	0
Reserved	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-48. Compare B High-Resolution Mirror Register (CMPBHRM) Field Descriptions

Bit	Field	Value	Description
15-8	CMPBHR	00-FFh	Compare B High-Resolution Bits Writes to both the CMPBHR and CMPBHRM locations access the high-resolution (least significant 8-bit) portion of the Counter Compare B value. The only difference is that unlike CMPBHR, reads from the mirror register, CMPBHRM, are indeterminate (reserved for TI Test). By default writes to this register are shadowed. Shadowing is enabled and disabled by the CMPCTL[SHDWBMODE] bit as described for the CMPBM register.
7-0	Reserved		Reserved for TI test.

Copyright © 2012–2019, Texas Instruments Incorporated

7.4.3 Action-Qualifier Submodule Registers

Figure 7-99 through Figure 7-102 and Table 7-49 through Table 7-52 provide the action-qualifier submodule register definitions.

Figure 7-99. Action-Qualifier Output A Control Register and Mirror Register (AQCTLA / AQCTLAM)

15	14	13	12	11	10	9	8
Reserved				CBD		CBU	
R-0				R/W-0		R/W-0	

7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-49. Action-Qualifier Output A Control Register and Mirror Register (AQCTLA / AQCTLAM) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved		Reserved
11-10	CBD		Action when the time-base counter equals the active CMPB register and the counter is decrementing.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
9-8	CBU		Action when the counter equals the active CMPB register and the counter is incrementing.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
7-6	CAD		Action when the counter equals the active CMPA register and the counter is decrementing.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
5-4	CAU		Action when the counter equals the active CMPA register and the counter is incrementing.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
3-2	PRD		Action when the counter equals the period. Note: By definition, in count up-down mode when the counter equals period the direction is defined as 0 or counting down.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-49. Action-Qualifier Output A Control Register and Mirror Register (AQCTLA / AQCTLAM) Field Descriptions (continued)

Bit	Field	Value	Description
1-0	ZRO		Action when counter equals zero.
			Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.

Figure 7-100. Action-Qualifier Output B Control Register and Mirror Register (AQCTLB / AQCTLBM)

15	12	11	10	9	8
Reserved		CBD		CBU	
R-0		R/W-0		R/W-0	

7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-50. Action-Qualifier Output B Control Register and Mirror Register (AQCTLB / AQCTLBM) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved		Reserved
11-10	CBD		Action when the counter equals the active CMPB register and the counter is decrementing.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.
9-8	CBU		Action when the counter equals the active CMPB register and the counter is incrementing.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.
7-6	CAD		Action when the counter equals the active CMPA register and the counter is decrementing.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.
5-4	CAU		Action when the counter equals the active CMPA register and the counter is incrementing.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-50. Action-Qualifier Output B Control Register and Mirror Register (AQCTLB / AQCTLBM) Field Descriptions (continued)

Bit	Field	Value	Description
3-2	PRD		Action when the counter equals the period.
			Note: By definition, in count up-down mode when the counter equals period the direction is defined as 0 or counting down.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.
1-0	ZRO		Action when counter equals zero.
			Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.

Figure 7-101. Action-Qualifier Software Force Register and Mirror Register (AQSFRC / AQSFRCM)

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
RLDCSF		OTSFB	ACTSFB		OTSFA	ACTSFA	
R/W-0		R/W-0	R/W-0		R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-51. Action-Qualifier Software Force Register and Mirror Register (AQSFRC / AQSFRCM) Field Descriptions

Bit	Field	Value	Description
15:8	Reserved		
7:6	RLDCSF		AQCSFRC Active Register Reload From Shadow Options
		00	Load on event counter equals zero
		01	Load on event counter equals period
		10	Load on event counter equals zero or counter equals period
		11	Load immediately (the active register is directly accessed by the CPU and is not loaded from the shadow register).
5	OTSFB		One-Time Software Forced Event on Output B
		0	Writing a 0 (zero) has no effect. Always reads back a 0
			This bit is auto cleared once a write to this register is complete, i.e., a forced event is initiated.)
			This is a one-shot forced event. It can be overridden by another subsequent event on output B.
		1	Initiates a single s/w forced event
4:3	ACTSFB		Action when One-Time Software Force B Is invoked
		00	Does nothing (action disabled)
		01	Clear (low)
		10	Set (high)
		11	Toggle (Low -> High, High -> Low)
			Note: This action is not qualified by counter direction (CNT_dir)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-51. Action-Qualifier Software Force Register and Mirror Register (AQSFRC / AQSFRCM) Field Descriptions (continued)

Bit	Field	Value	Description
2	OTSFA		One-Time Software Forced Event on Output A
		0	Writing a 0 (zero) has no effect. Always reads back a 0.
			This bit is auto cleared once a write to this register is complete (i.e., a forced event is initiated).
		1	Initiates a single software forced event
1:0	ACTSFA		Action When One-Time Software Force A Is Invoked
		00	Does nothing (action disabled)
		01	Clear (low)
		10	Set (high)
		11	Toggle (Low → High, High → Low)
			Note: This action is not qualified by counter direction (CNT_dir)

Figure 7-102. Action-Qualifier Continuous Software Force Register and Mirror Register (AQCSFRC / AQCSFRCM)

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		CSFB		CSFA	
R-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-52. Action-Qualifier Continuous Software Force Register and Mirror Register (AQCSFRC / AQCSFRCM) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3-2	CSFB		Continuous Software Force on Output B
			In immediate mode, a continuous force takes effect on the next TBCLK edge.
			In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register. To configure shadow mode, use AQSFRC[RLDCSF].
		00	Software forcing is disabled and has no effect
		01	Forces a continuous low on output B
		10	Forces a continuous high on output B
		11	Software forcing is disabled and has no effect
1-0	CSFA		Continuous Software Force on Output A
			In immediate mode, a continuous force takes effect on the next TBCLK edge.
			In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register.
		00	Software forcing is disabled and has no effect
		01	Forces a continuous low on output A
		10	Forces a continuous high on output A
		11	Software forcing is disabled and has no effect

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-103. Action Qualifier Control Register (AQCTLR)

15	12	11	10	9	8
Reserved		LDAQBSYNC		LDAQASYNC	
R-0		R/W-0		R/W-0	

7	6	5	4	3	2	1	0
Reserved	SHDWAQBMODE	Reserved	SHDWAQAMODE	LDAQBMODE		LDAQAMODE	
R-0	R/W-0	R-0	R/W-0	R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 7-53. Action Qualifier Control Register (AQCTLR) Field Description

Bit	Field	Value	Description
15-12	Reserved	0	Reserved
11-10	LDAQBSYNC		Shadow to Active AQCTLB Register Load on SYNC event
		00	Shadow to Active Load of AQCTLB occurs according to LDAQBMODE
		01	Shadow to Active Load of AQCTLB occurs both according to LDAQBMODE bits and when SYNC occurs.
		10	Shadow to Active Load of AQCTLB occurs only when a SYNC is received.
		11	Reserved
			Note: This bit is valid only if AQCTLR[SHDWAQBMODE] = 1.
9-8	LDAQASYNC		Shadow to Active AQCTLA Register Load on SYNC event
		00	Shadow to Active Load of AQCTLA occurs according to LDAQAMODE
		01	Shadow to Active Load of AQCTLA occurs both according to LDAQAMODE bits and when SYNC occurs.
		10	Shadow to Active Load of AQCTLA occurs only when a SYNC is received.
		11	Reserved
			Note: This bit is valid only if AQCTLR[SHDWAQAMODE] = 1.
7	Reserved	0	Reserved
6	SHDWAQBMODE		Action Qualifier B Register operating mode:
		1	Shadow mode – operates as a double buffer. All writes via the CPU access Shadow register.
		0	Immediate mode – only the Active action qualifier register is used. All writes/reads via the CPU directly access the Active register.
5	Reserved	0	Reserved
4	SHDWAQAMODE		Action Qualifier A Register operating mode:
		1	Shadow mode – operates as a double buffer. All writes via the CPU access Shadow register.
		0	Immediate mode – only the Active action qualifier register is used. All writes/reads via the CPU directly access the Active register.
3-2	LDAQBMODE		Active Action Qualifier B Load from Shadow Select Mode:
		00	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		01	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		10	Load on either CTR = Zero or CTR = PRD
		11	Freeze (no loads possible)
			Note: has no effect in Immediate mode.
1-0	LDAQAMODE		Active Action Qualifier A Load from Shadow Select Mode:
		00	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		01	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		10	Load on either CTR = Zero or CTR = PRD
		11	Freeze (no loads possible)
			Note: has no effect in Immediate mode.

Copyright © 2012–2019, Texas Instruments Incorporated

7.4.4 Dead-Band Submodule Registers

Figure 7-104 through Figure 7-106 and Table 7-54 through Table 7-56 provide the register definitions.

Figure 7-104. Dead-Band Generator Control Register (DBCTL)

15	14	13	12	11	10	9	8
HALFCYCLE	DEDB_MODE	OUTSWAP		SHDWDBFED MODE	SHDWDBRED MODE	LOADFEDMODE	
R/W-0							

7	6	5	4	3	2	1	0
LOADREDMODE		IN_MODE		POLSEL		OUT_MODE	
		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-54. Dead-Band Generator Control Register (DBCTL) Field Descriptions

Bit	Field	Value	Description
15	HALFCYCLE		Half Cycle Clocking Enable Bit
		0	Full cycle clocking enabled. The dead-band counters are clocked at the TBCLK rate.
		1	Half cycle clocking enabled. The dead-band counters are clocked at TBCLK*2.
14	DEDB_MODE		Dead Band Dual-Edge B Mode Control (S8 switch)
		0	Rising edge delay applied to InA/InB as selected by S4 switch (IN-MODE bits) on A signal path only. Falling edge delay applied to InA/InB as selected by S5 switch (IN-MODE bits) on B signal path only.
		1	Rising edge delay and falling edge delay applied to source selected by S4 switch (IN-MODE bits) and output to B signal path only. Note: When this bit is set to 1, user should always either set OUT_MODE bits such that Apath = InA OR OUTSWAP bits such that OutA=Bpath; otherwise, OutA will be invalid.
13-12	OUTSWAP		Dead Band Output Swap Control
			Bit 13 controls the S7 switch and bit 12 controls the S6 switch shown in Figure 7-31.
		00	OutA and OutB signals are as defined by OUT-MODE bits.
		01	OutA = A-path as defined by OUT-MODE bits.
			OutB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A-signal path).
		10	OutA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B-signal path).
			OutB = B-path as defined by OUT-MODE bits.
		11	OutA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B-signal path).
			OutB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A-signal path).
11	SHDWDBFEDMODE		FED Dead-Band Load Mode
		0	Immediate mode. Only the active DBFED register is used. All writes/reads via the CPU directly access the active register for immediate "FED dead-band action."
		1	Shadow mode. Operates as a double buffer. All writes via the CPU access Shadow register. Default at Reset is Immediate mode (for compatibility with legacy).
10	SHDWDBREDMODE		RED Dead Band Load mode
		0	Immediate mode. Only the active DBRED register is used. All writes/reads via the CPU directly access the active register for immediate "RED dead-band action."
		1	Shadow mode. Operates as a double buffer. All writes via the CPU access the Shadow register. Default at Reset is Immediate mode (for compatibility with legacy).

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-54. Dead-Band Generator Control Register (DBCTL) Field Descriptions (continued)

Bit	Field	Value	Description
9-8	LOADFEDMODE		Active DBFED Load from Shadow Select Mode
		00	Load on CTR = 0
		01	Load on CTR = PRD
		10	Load on either CTR = 0 , or CTR = PRD
		11	Freeze (no loads possible)
			Note: has no effect in Immediate mode.
7-6	LOADREDMODE		Active DBRED Load from Shadow Select Mode
		00	Load on Counter = 0 (CNT_eq)
		01	Load on Counter = Period (PRD_eq)
		10	Load on either Counter = 0, or Counter = Period
		11	Freeze (no loads possible)
			Note: has no effect in Immediate mode.
5-4	IN_MODE		Dead-Band Input Mode Control
			Bit 5 controls the S5 switch and bit 4 controls the S4 switch shown in Figure 7-31.
			This allows you to select the input source to the falling-edge and rising-edge delay.
			To produce classical dead-band waveforms the default is EPWMxA In is the source for both falling and rising-edge delays.
		00	EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.
		01	EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal.
			EPWMxA In (from the action-qualifier) is the source for falling-edge delayed signal.
		10	EPWMxA In (from the action-qualifier) is the source for rising-edge delayed signal.
			EPWMxB In (from the action-qualifier) is the source for falling-edge delayed signal.
		11	EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.
3-2	POLSEL		Polarity Select Control
			Bit 3 controls the S3 switch and bit 2 controls the S2 switch shown in Figure 7-31.
			This allows you to selectively invert one of the delayed signals before it is sent out of the dead-band submodule.
			The following descriptions correspond to classical upper/lower switch control as found in one leg of a digital motor control inverter.
			These assume that DBCTL[OUT_MODE] = 1,1 and DBCTL[IN_MODE] = 0,0. Other enhanced modes are also possible, but not regarded as typical usage modes.
		00	Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).
		01	Active low complementary (ALC) mode. EPWMxA is inverted.
		10	Active high complementary (AHC). EPWMxB is inverted.
		11	Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.
1-0	OUT_MODE		Dead-Band Output Mode Control
			Bit 1 controls the S1 switch and bit 0 controls the S0 switch shown in Figure 7-31.
		00	DBM is fully disabled or by-passed. In this mode the POLSEL and IN-MODE bits have no effect.
		01	Apath = InA (delay is by-passed for A signal path)
			Bpath = FED (Falling Edge Delay in B signal path)
		10	Apath = RED (Rising Edge Delay in A signal path)
			Bpath = InB (delay is by-passed for B signal path)
		11	DBM is fully enabled (i.e. both RED and FED active)

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-105. Dead-Band Generator Rising Edge Delay and Mirror Register (DBRED / DBREDM)

15	14	13	8
Reserved		DEL	
R-0		R/W-0	

7	0
DEL	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-55. Dead-Band Generator Rising Edge Delay and Mirror Register (DBRED / DBREDM) Field Descriptions

Bit	Field	Value	Description
15-14	Reserved	0	Reserved
13-0	DEL		Rising Edge Delay Count. 14-bit counter.

Figure 7-106. Dead-Band Generator Falling Edge Delay and Mirror Register (DBFED / DBFEDM)

15	14	13	8
Reserved		DEL	
R-0		R/W-0	

7	0
DEL	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-56. Dead-Band Generator Falling Edge Delay and Mirror Register (DBFED / DBFEDM) Field Descriptions

Bit	Field	Value	Description
15-14	Reserved		Reserved
13-0	DEL		Falling Edge Delay Count. 14-bit counter

Figure 7-107. Dead Band Rising Edge Delay High-Resolution Register (DBREDHR)

15	9	8	0
DBREDHR		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-57. Dead Band Rising Edge Delay High-Resolution Register (DBREDHR) Field Descriptions

Bit	Field	Value	Description
15-9	DBREDHR	00-7Fh	These 7-bits contain the high-resolution portion (least significant 7-bits) of the dead-band rising edge delay value. DBREDM: DBREDHR can be accessed in a single 32-bit read/write. Shadowing is enabled and disabled by the DBCTL[SHDWDBREDMODE] bit as described for the DBRED register.
8-0	Reserved	0	Reserved for TI Test

Figure 7-108. Dead Band Falling Edge Delay High-Resolution Register (DBFEDHR)

15	9	8	0
DBFEDHR		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-58. Dead Band Falling Edge Delay High-Resolution Register (DBFEDHR) Field Descriptions

Bit	Field	Value	Description
15-9	DBFEDHR	00-7Fh	These 7-bits contain the high-resolution portion (least significant 7-bits) of the dead-band rising edge delay value. DBFEDM: DBFEDHR can be accessed in a single 32-bit read/write. Shadowing is enabled and disabled by the DBCTL[SHDWDBFEDMODE] bit as described for the DBFED register.
8-0	Reserved	0	Reserved for TI Test

Copyright © 2012–2019, Texas Instruments Incorporated

7.4.5 PWM-Chopper Submodule Control Register

Figure 7-109 and Table 7-59 provide the definitions for the PWM-chopper submodule control register.

Figure 7-109. PWM-Chopper Control Register (PCCTL)

15	11	10	8
Reserved		CHPDUTY	
R-0		R/W-0	

7	5	4	1	0
CHPFREQ		OSHTWTH		CHPEN
R/W-0		R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-59. PWM-Chopper Control Register (PCCTL) Bit Descriptions

Bit	Field	Value	Description
15-11	Reserved		Reserved
10-8	CHPDUTY		Chopping Clock Duty Cycle
		000	Duty = 1/8 (12.5%)
		001	Duty = 2/8 (25.0%)
		010	Duty = 3/8 (37.5%)
		011	Duty = 4/8 (50.0%)
		100	Duty = 5/8 (62.5%)
		101	Duty = 6/8 (75.0%)
		110	Duty = 7/8 (87.5%)
		111	Reserved
7:5	CHPFREQ		Chopping Clock Frequency
		000	Divide by 1 (no prescale, = 12.5 MHz at 100 MHz SYSCLKOUT)
		001	Divide by 2 (6.25 MHz at 100 MHz SYSCLKOUT)
		010	Divide by 3 (4.16 MHz at 100 MHz SYSCLKOUT)
		011	Divide by 4 (3.12 MHz at 100 MHz SYSCLKOUT)
		100	Divide by 5 (2.50 MHz at 100 MHz SYSCLKOUT)
		101	Divide by 6 (2.08 MHz at 100 MHz SYSCLKOUT)
		110	Divide by 7 (1.78 MHz at 100 MHz SYSCLKOUT)
		111	Divide by 8 (1.56 MHz at 100 MHz SYSCLKOUT)
4:1	OSHTWTH		One-Shot Pulse Width
		0000	1 x SYSCLKOUT / 8 wide (= 80 nS at 100 MHz SYSCLKOUT)
		0001	2 x SYSCLKOUT / 8 wide (= 160 nS at 100 MHz SYSCLKOUT)
		0010	3 x SYSCLKOUT / 8 wide (= 240 nS at 100 MHz SYSCLKOUT)
		0011	4 x SYSCLKOUT / 8 wide (= 320 nS at 100 MHz SYSCLKOUT)
		0100	5 x SYSCLKOUT / 8 wide (= 400 nS at 100 MHz SYSCLKOUT)
		0101	6 x SYSCLKOUT / 8 wide (= 480 nS at 100 MHz SYSCLKOUT)
		0110	7 x SYSCLKOUT / 8 wide (= 560 nS at 100 MHz SYSCLKOUT)
		0111	8 x SYSCLKOUT / 8 wide (= 640 nS at 100 MHz SYSCLKOUT)
		1000	9 x SYSCLKOUT / 8 wide (= 720 nS at 100 MHz SYSCLKOUT)
		1001	10 x SYSCLKOUT / 8 wide (= 800 nS at 100 MHz SYSCLKOUT)
		1010	11 x SYSCLKOUT / 8 wide (= 880 nS at 100 MHz SYSCLKOUT)
		1011	12 x SYSCLKOUT / 8 wide (= 960 nS at 100 MHz SYSCLKOUT)
		1100	13 x SYSCLKOUT / 8 wide (= 1040 nS at 100 MHz SYSCLKOUT)
		1101	14 x SYSCLKOUT / 8 wide (= 1120 nS at 100 MHz SYSCLKOUT)
		1110	15 x SYSCLKOUT / 8 wide (= 1200 nS at 100 MHz SYSCLKOUT)
		1111	16 x SYSCLKOUT / 8 wide (= 1280 nS at 100 MHz SYSCLKOUT)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-59. PWM-Chopper Control Register (PCCTL) Bit Descriptions (continued)

Bit	Field	Value	Description
0	CHPEN		PWM-chopping Enable
		0	Disable (bypass) PWM chopping function
		1	Enable chopping function

Copyright © 2012–2019, Texas Instruments Incorporated

7.4.6 Trip-Zone Submodule Control and Status Registers

Figure 7-110. Trip-Zone Select Register (TZSEL)

15	14	13	12	11	10	9	8
DCBEVT1	DCAEVT1	OSHT6	OSHT5	OSHT4	OSHT3	OSHT2	OSHT1
R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DCBEVT2	DCAEVT2	CBC6	CBC5	CBC4	CBC3	CBC2	CBC1
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-60. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions

Bit	Field	Value	Description
One-Shot (OSHT) Trip-zone enable/disable. When any of the enabled pins go low, a one-shot trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register (Figure 7-111) is taken on the EPWMxA and EPWMxB outputs. The one-shot trip condition remains latched until the user clears the condition via the TZCLR register (Figure 7-114).			
15	DCBEVT1		Digital Compare Output B Event 1 Select
		0	Disable DCBEVT1 as one-shot-trip source for this ePWM module.
		1	Enable DCBEVT1 as one-shot-trip source for this ePWM module.
14	DCAEVT1		Digital Compare Output A Event 1 Select
		0	Disable DCAEVT1 as one-shot-trip source for this ePWM module.
		1	Enable DCAEVT1 as one-shot-trip source for this ePWM module.
13	OSHT6		Trip-zone 6 ($\overline{TZ6}$) Select
		0	Disable $\overline{TZ6}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZ6}$ as a one-shot trip source for this ePWM module.
12	OSHT5		Trip-zone 5 ($\overline{TZ5}$) Select
		0	Disable $\overline{TZ5}$ as a one-shot trip source for this ePWM module
		1	Enable $\overline{TZ5}$ as a one-shot trip source for this ePWM module
11	OSHT4		Trip-zone 4 ($\overline{TZ4}$) Select
		0	Disable $\overline{TZ4}$ as a one-shot trip source for this ePWM module
		1	Enable $\overline{TZ4}$ as a one-shot trip source for this ePWM module
10	OSHT3		Trip-zone 3 ($\overline{TZ3}$) Select
		0	Disable $\overline{TZ3}$ as a one-shot trip source for this ePWM module
		1	Enable $\overline{TZ3}$ as a one-shot trip source for this ePWM module
9	OSHT2		Trip-zone 2 ($\overline{TZ2}$) Select
		0	Disable $\overline{TZ2}$ as a one-shot trip source for this ePWM module
		1	Enable $\overline{TZ2}$ as a one-shot trip source for this ePWM module
8	OSHT1		Trip-zone 1 ($\overline{TZ1}$) Select
		0	Disable $\overline{TZ1}$ as a one-shot trip source for this ePWM module
		1	Enable $\overline{TZ1}$ as a one-shot trip source for this ePWM module
Cycle-by-Cycle (CBC) Trip-zone enable/disable. When any of the enabled pins go low, a cycle-by-cycle trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register (Figure 7-111) is taken on the EPWMxA and EPWMxB outputs. A cycle-by-cycle trip condition is automatically cleared when the time-base counter reaches zero.			
7	DCBEVT2		Digital Compare Output B Event 2 Select
		0	Disable DCBEVT2 as a CBC trip source for this ePWM module
		1	Enable DCBEVT2 as a CBC trip source for this ePWM module
6	DCAEVT2		Digital Compare Output A Event 2 Select
		0	Disable DCAEVT2 as a CBC trip source for this ePWM module
		1	Enable DCAEVT2 as a CBC trip source for this ePWM module

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-60. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions (continued)

Bit	Field	Value	Description
5	CBC6		Trip-zone 6 ($\overline{TZ6}$) Select
		0	Disable $\overline{TZ6}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ6}$ as a CBC trip source for this ePWM module
4	CBC5		Trip-zone 5 ($\overline{TZ5}$) Select
		0	Disable $\overline{TZ5}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ5}$ as a CBC trip source for this ePWM module
3	CBC4		Trip-zone 4 ($\overline{TZ4}$) Select
		0	Disable $\overline{TZ4}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ4}$ as a CBC trip source for this ePWM module
2	CBC3		Trip-zone 3 ($\overline{TZ3}$) Select
		0	Disable $\overline{TZ3}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ3}$ as a CBC trip source for this ePWM module
1	CBC2		Trip-zone 2 ($\overline{TZ2}$) Select
		0	Disable $\overline{TZ2}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ2}$ as a CBC trip source for this ePWM module
0	CBC1		Trip-zone 1 ($\overline{TZ1}$) Select
		0	Disable $\overline{TZ1}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ1}$ as a CBC trip source for this ePWM module

Figure 7-111. Trip-Zone Control Register (TZCTL)

15 12	11 10	9 8
Reserved	DCBEVT2	DCBEVT1
R-0	R/W-0	R/W-0

7 6	5 4	3 2	1 0
DCAEVT2	DCAEVT1	TZB	TZA
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-61. Trip-Zone Control Register Field Descriptions

Bit	Field	Value	Description
15-12	Reserved		Reserved
11-10	DCBEVT2		Digital Compare Output B Event 2 Action On EPWMxB:
		00	High-impedance (EPWMxB = High-impedance state)
		01	Force EPWMxB to a high state.
		10	Force EPWMxB to a low state.
		11	Do Nothing, trip action is disabled
9-8	DCBEVT1		Digital Compare Output B Event 1 Action On EPWMxB:
		00	High-impedance (EPWMxB = High-impedance state)
		01	Force EPWMxB to a high state.
		10	Force EPWMxB to a low state.
		11	Do Nothing, trip action is disabled
7-6	DCAEVT2		Digital Compare Output A Event 2 Action On EPWMxA:
		00	High-impedance (EPWMxA = High-impedance state)
		01	Force EPWMxA to a high state.
		10	Force EPWMxA to a low state.
		11	Do Nothing, trip action is disabled

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-61. Trip-Zone Control Register Field Descriptions (continued)

Bit	Field	Value	Description
5-4	DCAEVT1		Digital Compare Output A Event 1 Action On EPWMxA:
		00	High-impedance (EPWMxA = High-impedance state)
		01	Force EPWMxA to a high state.
		10	Force EPWMxA to a low state.
		11	Do Nothing, trip action is disabled
3-2	TZB		When a trip event occurs the following action is taken on output EPWMxB. Which trip-zone pins can cause an event is defined in the TZSEL register.
		00	High-impedance (EPWMxB = High-impedance state)
		01	Force EPWMxB to a high state
		10	Force EPWMxB to a low state
		11	Do nothing, no action is taken on EPWMxB.
1-0	TZA		When a trip event occurs the following action is taken on output EPWMxA. Which trip-zone pins can cause an event is defined in the TZSEL register.
		00	High-impedance (EPWMxA = High-impedance state)
		01	Force EPWMxA to a high state
		10	Force EPWMxA to a low state
		11	Do nothing, no action is taken on EPWMxA.

Figure 7-112. Trip-Zone Enable Interrupt Register (TZEINT)

15							8
Reserved							
R -0							

7	6	5	4	3	2	1	0
Reserved	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	Reserved
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-62. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions

Bit	Field	Value	Description
15-3	Reserved		Reserved
6	DCBEVT2		Digital Comparator Output B Event 2 Interrupt Enable
		0	Disabled
		1	Enabled
5	DCBEVT1		Digital Comparator Output B Event 1 Interrupt Enable
		0	Disabled
		1	Enabled
4	DCAEVT2		Digital Comparator Output A Event 2 Interrupt Enable
		0	Disabled
		1	Enabled
3	DCAEVT1		Digital Comparator Output A Event 1 Interrupt Enable
		0	Disabled
		1	Enabled
2	OST		Trip-zone One-Shot Interrupt Enable
		0	Disable one-shot interrupt generation
		1	Enable Interrupt generation; a one-shot trip event will cause a EPWMx_TZINT PIE interrupt.
1	CBC		Trip-zone Cycle-by-Cycle Interrupt Enable
		0	Disable cycle-by-cycle interrupt generation.
		1	Enable interrupt generation; a cycle-by-cycle trip event will cause an EPWMx_TZINT PIE interrupt.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-62. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions (continued)

Bit	Field	Value	Description
0	Reserved		Reserved

Figure 7-113. Trip-Zone Flag Register (TZFLG)

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	INT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 7-63. Trip-Zone Flag Register (TZFLG) Field Descriptions

Bit	Field	Value	Description
15:7	Reserved		Reserved
6	DCBEVT2		Latched Status Flag for Digital Compare Output B Event 2
		0	Indicates no trip event has occurred on DCBEVT2
		1	Indicates a trip event has occurred for the event defined for DCBEVT2
5	DCBEVT1		Latched Status Flag for Digital Compare Output B Event 1
		0	Indicates no trip event has occurred on DCBEVT1
		1	Indicates a trip event has occurred for the event defined for DCBEVT1
4	DCAEVT2		Latched Status Flag for Digital Compare Output A Event 2
		0	Indicates no trip event has occurred on DCAEVT2
		1	Indicates a trip event has occurred for the event defined for DCAEVT2
3	DCAEVT1		Latched Status Flag for Digital Compare Output A Event 1
		0	Indicates no trip event has occurred on DCAEVT1
		1	Indicates a trip event has occurred for the event defined for DCAEVT1
2	OST		Latched Status Flag for A One-Shot Trip Event
		0	No one-shot trip event has occurred.
		1	Indicates a trip event has occurred on a pin selected as a one-shot trip source.
			This bit is cleared by writing the appropriate value to the TZCLR register.
1	CBC		Latched Status Flag for Cycle-By-Cycle Trip Event
		0	No cycle-by-cycle trip event has occurred.
		1	Indicates a trip event has occurred on a signal selected as a cycle-by-cycle trip source. The TZFLG[CBC] bit will remain set until it is manually cleared by the user. If the cycle-by-cycle trip event is still present when the CBC bit is cleared, then CBC will be immediately set again. The specified condition on the signal is automatically cleared when the ePWM time-base counter reaches zero (TBCTR = 0x00) if the trip condition is no longer present. The condition on the signal is only cleared when the TBCTR = 0x00 no matter where in the cycle the CBC flag is cleared.
			This bit is cleared by writing the appropriate value to the TZCLR register.
0	INT		Latched Trip Interrupt Status Flag
		0	Indicates no interrupt has been generated.
		1	Indicates an EPWMx_TZINT PIE interrupt was generated because of a trip condition.
			No further EPWMx_TZINT PIE interrupts will be generated until this flag is cleared. If the interrupt flag is cleared when either CBC or OST is set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts.
			This bit is cleared by writing the appropriate value to the TZCLR register.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-114. Trip-Zone Clear Register and Mirror Register (TZCLR / TZCLRM)

15	14	13	8
CBCPULSE		Reserved	
R/W-0		R-0	

7	6	5	4	3	2	1	0
Reserved	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	INT
R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; *n*C - write *n* to clear; R = Read only; -*n* = value after reset

Table 7-64. Trip-Zone Clear Register and Mirror Register (TZCLR / TZCLRM) Field Descriptions

Bit	Field	Value	Description
15-14	CBCPULSE		Clear Pulse for Cycle-By-Cycle (CBC) Trip Latch
			This bit field determines which pulse clears the CBC trip latch.
		00	CTR = zero pulse clears CBC trip latch. (Same as legacy designs.)
		01	CTR = PRD pulse clears CBC trip latch.
		10	CTR = zero or CTR = PRD pulse clears CBC trip latch.
		11	CBC trip latch is not cleared
13-7	Reserved		Reserved
6	DCBEVT2		Clear Flag for Digital Compare Output B Event 2
		0	Writing 0 has no effect. This bit always reads back 0.
		1	Writing 1 clears the DCBEVT2 event trip condition.
5	DCBEVT1		Clear Flag for Digital Compare Output B Event 1
		0	Writing 0 has no effect. This bit always reads back 0.
		1	Writing 1 clears the DCBEVT1 event trip condition.
4	DCAEVT2		Clear Flag for Digital Compare Output A Event 2
		0	Writing 0 has no effect. This bit always reads back 0.
		1	Writing 1 clears the DCAEVT2 event trip condition.
3	DCAEVT1		Clear Flag for Digital Compare Output A Event 1
		0	Writing 0 has no effect. This bit always reads back 0.
		1	Writing 1 clears the DCAEVT1 event trip condition.
2	OST		Clear Flag for One-Shot Trip (OST) Latch
		0	Has no effect. Always reads back a 0.
		1	Clears this Trip (set) condition.
1	CBC		Clear Flag for Cycle-By-Cycle (CBC) Trip Latch
		0	Has no effect. Always reads back a 0.
		1	Clears this Trip (set) condition.
0	INT		Global Interrupt Clear Flag
		0	Has no effect. Always reads back a 0.
		1	Clears the trip-interrupt flag for this ePWM module (TZFLG[INT]).
			NOTE: No further EPWMx_TZINT PIE interrupts will be generated until the flag is cleared. If the TZFLG[INT] bit is cleared and any of the other flag bits are set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-115. Trip-Zone Force Register (TZFRC)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	Reserved
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R- 0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-65. Trip-Zone Force Register (TZFRC) Field Descriptions

Bit	Field	Value	Description
15- 7	Reserved		Reserved
6	DCBEVT2		Force Flag for Digital Compare Output B Event 2
		0	Writing 0 has no effect. This bit always reads back 0.
		1	Writing 1 forces the DCBEVT2 event trip condition and sets the TZFLG[DCBEVT2] bit.
5	DCBEVT1		Force Flag for Digital Compare Output B Event 1
		0	Writing 0 has no effect. This bit always reads back 0.
		1	Writing 1 forces the DCBEVT1 event trip condition and sets the TZFLG[DCBEVT1] bit.
4	DCAEVT2		Force Flag for Digital Compare Output A Event 2
		0	Writing 0 has no effect. This bit always reads back 0.
		1	Writing 1 forces the DCAEVT2 event trip condition and sets the TZFLG[DCAEVT2] bit.
3	DCAEVT1		Force Flag for Digital Compare Output A Event 1
		0	Writing 0 has no effect. This bit always reads back 0
		1	Writing 1 forces the DCAEVT1 event trip condition and sets the TZFLG[DCAEVT1] bit.
2	OST		Force a One-Shot Trip Event via Software
		0	Writing of 0 is ignored. Always reads back a 0.
		1	Forces a one-shot trip event and sets the TZFLG[OST] bit.
1	CBC		Force a Cycle-by-Cycle Trip Event via Software
		0	Writing of 0 is ignored. Always reads back a 0.
		1	Forces a cycle-by-cycle trip event and sets the TZFLG[CBC] bit.
0	Reserved		Reserved

Figure 7-116. Trip-Zone Digital Compare Event Select Register (TZDCSEL)

15 12	11 9	8 6	5 3	2 0
Reserved	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1
R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-66. Trip Zone Digital Compare Event Select Register (TZDCSEL) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved		Reserved
11-9	DCBEVT2		Digital Compare Output B Event 2 Selection
		000	Event disabled
		001	DCBH = low, DCBL = don't care
		010	DCBH = high, DCBL = don't care
		011	DCBL = low, DCBH = don't care
		100	DCBL = high, DCBH = don't care
		101	DCBL = high, DCBH = low
		110	reserved
		111	reserved
8-6	DCBEVT1		Digital Compare Output B Event 1 Selection
		000	Event disabled
		001	DCBH = low, DCBL = don't care
		010	DCBH = high, DCBL = don't care
		011	DCBL = low, DCBH = don't care
		100	DCBL = high, DCBH = don't care
		101	DCBL = high, DCBH = low
		110	reserved
		111	reserved
5-3	DCAEVT2		Digital Compare Output A Event 2 Selection
		000	Event disabled
		001	DCAH = low, DCAL = don't care
		010	DCAH = high, DCAL = don't care
		011	DCAL = low, DCAH = don't care
		100	DCAL = high, DCAH = don't care
		101	DCAL = high, DCAH = low
		110	reserved
		111	reserved
2-0	DCAEVT1		Digital Compare Output A Event 1 Selection
		000	Event disabled
		001	DCAH = low, DCAL = don't care
		010	DCAH = high, DCAL = don't care
		011	DCAL = low, DCAH = don't care
		100	DCAL = high, DCAH = don't care
		101	DCAL = high, DCAH = low
		110	reserved
		111	reserved

Copyright © 2012–2019, Texas Instruments Incorporated

7.4.7 Digital Compare Submodule Registers

Figure 7-117. Digital Compare Trip Select (DCTRIPSEL)

15	12	11	8
DCBLCOMPSEL		DCBHCOMPSEL	
R/W-0		R/W-0	

7	4	3	0
DCALCOMPSEL		DCAHCOMPSEL	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-67. Digital Compare Trip Select (DCTRIPSEL) Field Descriptions

Bit	Field	Value	Description
15-12	DCBLCOMPSEL		Digital Compare B Low Input Select Bits
		0000	TRIPIN1 and ($\overline{TZ1}$ input)
		0001	TRIPIN2 and ($\overline{TZ2}$ input)
		0010	TRIPIN3 and ($\overline{TZ3}$ input)
		0011	TRIPIN4
		. . .	. . .
		1011	TRIPIN12
		1100	Reserved
		1101	TRIPIN14
		1110	TRIPIN15
		1111	Trip combination input (all trip inputs selected by DCBLTRIPSEL register ORed together)
11-8	DCBHCOMPSEL		Digital Compare B High Input Select Bits
		0000	TRIPIN1 and ($\overline{TZ1}$ input)
		0001	TRIPIN2 and ($\overline{TZ2}$ input)
		0010	TRIPIN3 and ($\overline{TZ3}$ input)
		0011	TRIPIN4
		. . .	. . .
		1011	TRIPIN12
		1100	Reserved
		1101	TRIPIN14
		1110	TRIPIN15
		1111	Trip combination input (all trip inputs selected by DCBHTRIPSEL register ORed together)
7-4	DCALCOMPSEL		Digital Compare A Low Input Select Bits
		0000	TRIPIN1 and ($\overline{TZ1}$ input)
		0001	TRIPIN2 and ($\overline{TZ2}$ input)
		0010	TRIPIN3 and ($\overline{TZ3}$ input)
		0011	TRIPIN4
		. . .	. . .
		1011	TRIPIN12
		1100	Reserved
		1101	TRIPIN14
		1110	TRIPIN15
		1111	Trip combination input (all trip inputs selected by DCALTRIPSEL register ORed together)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-67. Digital Compare Trip Select (DCTRIPSEL) Field Descriptions (continued)

Bit	Field	Value	Description
3-0	DCAHCOMPSEL		Digital Compare A High Input Select Bits
		0000	TRIPIN1 and ($\overline{TZ1}$ input)
		0001	TRIPIN2 and ($\overline{TZ2}$ input)
		0010	TRIPIN3 and ($\overline{TZ3}$ input)
		0011	TRIPIN4
		. . .	. . .
		1011	TRIPIN12
		1100	Reserved
		1101	TRIPIN14
		1110	TRIPIN15
		1111	Trip combination input (all trip inputs selected by DCAHTRIPSEL register ORed together)

Figure 7-118. Digital Compare A Control Register (DCACTL)

15	10	9	8
Reserved		EVT2FRC SYNCSEL	EVT2SRCSEL
R-0		R/W-0	R/W-0

7	4	3	2	1	0
Reserved		EVT1SYNCE	EVT1SOCE	EVT1FRC SYNCSEL	EVT1SRCSEL
R-0		R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-68. Digital Compare A Control Register (DCACTL) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved		Reserved
9	EVT2FRC SYNCSEL		DCAEVT2 Force Synchronization Signal Select
		0	Source Is Synchronous Signal
		1	Source Is Asynchronous Signal
8	EVT2SRCSEL		DCAEVT2 Source Signal Select
		0	Source Is DCAEVT2 Signal
		1	Source Is DCEVTFILT Signal
7-4	Reserved		Reserved
3	EVT1SYNCE		DCAEVT1 SYNC, Enable/Disable
		0	SYNC Generation Disabled
		1	SYNC Generation Enabled
2	EVT1SOCE		DCAEVT1 SOC, Enable/Disable
		0	SOC Generation Disabled
		1	SOC Generation Enabled
1	EVT1FRC SYNCSEL		DCAEVT1 Force Synchronization Signal Select
		0	Source Is Synchronous Signal
		1	Source Is Asynchronous Signal
0	EVT1SRCSEL		DCAEVT1 Source Signal Select
		0	Source Is DCAEVT1 Signal
		1	Source Is DCEVTFILT Signal

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-119. Digital Compare B Control Register (DCBCTL)

15	10	9	8
Reserved		EVT2FRC SYNCSEL	EVT2SRCSEL
R-0		R/W-0	R/W-0

7	4	3	2	1	0
Reserved		EVT1SYNCE	EVT1SOCE	EVT1FRC SYNCSEL	EVT1SRCSEL
R-0		R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-69. Digital Compare B Control Register (DCBCTL) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved		Reserved
9	EVT2FRC SYNCSEL		DCBEVT2 Force Synchronization Signal Select
		0	Source Is Synchronous Signal
		1	Source Is Asynchronous Signal
8	EVT2SRCSEL		DCBEVT2 Source Signal Select
		0	Source Is DCBEVT2 Signal
		1	Source Is DCEVTFILT Signal
7-4	Reserved		Reserved
3	EVT1SYNCE		DCBEVT1 SYNC, Enable/Disable
		0	SYNC Generation Disabled
		1	SYNC Generation Enabled
2	EVT1SOCE		DCBEVT1 SOC, Enable/Disable
		0	SOC Generation Disabled
		1	SOC Generation Enabled
1	EVT1FRC SYNCSEL		DCBEVT1 Force Synchronization Signal Select
		0	Source Is Synchronous Signal
		1	Source Is Asynchronous Signal
0	EVT1SRCSEL		DCBEVT1 Source Signal Select
		0	Source Is DCBEVT1 Signal
		1	Source Is DCEVTFILT Signal

Figure 7-120. Digital Compare Filter Control Register (DCFCTL)

15	13	12	8
Reserved		Reserved	
R-0		R-0	

7	6	5	4	3	2	1	0
Reserved	Reserved	PULSESEL		BLANKINV	BLANKE	SRCSEL	
R-0	R-0	R/W-0		R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-70. Digital Compare Filter Control Register (DCFCTL) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved		Reserved
12-8	Reserved		Reserved for TI Test
7	Reserved		Reserved
6	Reserved		Reserved for TI Test

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-70. Digital Compare Filter Control Register (DCFCTL) Field Descriptions (continued)

Bit	Field	Value	Description
5-4	PULSESEL		Pulse Select For Blanking & Capture Alignment
		00	Time-base counter equal to period (TBCTR = TBPRD)
		01	Time-base counter equal to zero (TBCTR = 0x00)
		10	Reserved
		11	Reserved
3	BLANKINV		Blanking Window Inversion
		0	Blanking window not inverted
		1	Blanking window inverted
2	BLANKE		Blanking Window Enable/Disable
		0	Blanking window is disabled
		1	Blanking window is enabled
1-0	SRCSEL		Filter Block Signal Source Select
		00	Source Is DCAEVT1 Signal
		01	Source Is DCAEVT2 Signal
		10	Source Is DCBEVT1 Signal
		11	Source Is DCBEVT2 Signal

Figure 7-121. Digital Compare Capture Control Register (DCCAPCTL)

15	8
Reserved	
R-0	

7	2	1	0
Reserved		SHDWMODE	CAPE
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-71. Digital Compare Capture Control Register (DCCAPCTL) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved		Reserved
1	SHDWMODE		TBCTR Counter Capture Shadow Select Mode
		0	Enable shadow mode. The DCCAP active register is copied to shadow register on a TBCTR = TBPRD or TBCTR = zero event as defined by the DCFCTL[PULSESEL] bit. CPU reads of the DCCAP register will return the shadow register contents.
		1	Active Mode. In this mode the shadow register is disabled. CPU reads from the DCCAP register will always return the active register contents.
0	CAPE		TBCTR Counter Capture Enable/Disable
		0	Disable the time-base counter capture.
		1	Enable the time-base counter capture.

Figure 7-122. Digital Compare Counter Capture Register (DCCAP)

15	0
DCCAP	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-72. Digital Compare Counter Capture Register (DCCAP) Field Descriptions

Bit	Field	Value	Description
15-0	DCCAP	0000-FFFFh	Digital Compare Time-Base Counter Capture To enable time-base counter capture, set the DCCAPCLT[CAPE] bit to 1. If enabled, reflects the value of the time-base counter (TBCTR) on the low to high edge transition of a filtered (DCEVTFLT) event. Further capture events are ignored until the next period or zero as selected by the DCFCTL[PULSESEL] bit. Shadowing of DCCAP is enabled and disabled by the DCCAPCTL[SHDWMODE] bit. By default this register is shadowed. • If DCCAPCTL[SHDWMODE] = 0, then the shadow is enabled. In this mode, the active register is copied to the shadow register on the TBCTR = TBPRD or TBCTR = zero as defined by the DCFCTL[PULSESEL] bit. CPU reads of this register will return the shadow register value. • If DCCAPCTL[SHDWMODE] = 1, then the shadow register is disabled. In this mode, CPU reads will return the active register value. The active and shadow registers share the same memory map address.

Figure 7-123. Digital Compare Filter Offset Register (DCFOFFSET)

15	0
DCOFFSET	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-73. Digital Compare Filter Offset Register (DCFOFFSET) Field Descriptions

Bit	Field	Value	Description
15-0	OFFSET	0000- FFFFh	Blanking Window Offset These 16-bits specify the number of TBCLK cycles from the blanking window reference to the point when the blanking window is applied. The blanking window reference is either period or zero as defined by the DCFCTL[PULSESEL] bit. This offset register is shadowed and the active register is loaded at the reference point defined by DCFCTL[PULSESEL]. The offset counter is also initialized and begins to count down when the active register is loaded. When the counter expires, the blanking window is applied. If the blanking window is currently active, then the blanking window counter is restarted.

Figure 7-124. Digital Compare Filter Offset Counter Register (DCFOFFSETCNT)

15	0
OFFSETCNT	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-74. Digital Compare Filter Offset Counter Register (DCFOFFSETCNT) Field Descriptions

Bit	Field	Value	Description
15-0	OFFSETCNT	0000- FFFFh	Blanking Offset Counter These 16-bits are read only and indicate the current value of the offset counter. The counter counts down to zero and then stops until it is re-loaded on the next period or zero event as defined by the DCFCTL[PULSESEL] bit. The offset counter is not affected by the free/soft emulation bits. That is, it will always continue to count down if the device is halted by a emulation stop.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-125. Digital Compare Filter Window Register (DCFWINDOW)

15							8
Reserved							
R-0							
7							0
WINDOW							
R/W-0							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-75. Digital Compare Filter Window Register (DCFWINDOW) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved		Reserved
7-0	WINDOW		Blanking Window Width
		00h	No blanking window is generated.
		01-FFh	Specifies the width of the blanking window in TBCLK cycles. The blanking window begins when the offset counter expires. When this occurs, the window counter is loaded and begins to count down. If the blanking window is currently active and the offset counter expires, the blanking window counter is restarted.
			The blanking window can cross a PWM period boundary.

Figure 7-126. Digital Compare Filter Window Counter Register (DCFWINDOWCNT)

15							8
Reserved							
R-0							
7							0
WINDOWCNT							
R-0							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-76. Digital Compare Filter Window Counter Register (DCFWINDOWCNT) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved		Any writes to these bit(s) must always have a value of 0.
7-0	WINDOWCNT	00-FF	Blanking Window Counter
			These 8 bits are read only and indicate the current value of the window counter. The counter counts down to zero and then stops until it is re-loaded when the offset counter reaches zero again.

Figure 7-127. Digital Compare A High Trip Input Select (DCAHTRIPSEL) (EALLOW-protected)

15	14	13	12	11	10	9	8
Reserved	TRIPIN15	TRIPIN14	Reserved	TRIPIN12	TRIPIN11	TRIPIN10	TRIPIN9
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TRIPIN8	TRIPIN7	TRIPIN6	TRIPIN5	TRIPIN4	TRIPIN3	TRIPIN2	TRIPIN1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-77. Digital Compare A High Trip Input Select (DCAHTRIPSEL) Field Descriptions

Bit	Field	Value	Description
15	Reserved		Reserved
14	TRIPIN15		TRIP Input 15
		0	Trip Input 15 not selected as combinational ORed input
		1	Trip Input 15 selected as combinational ORed input to DCAH mux
13	TRIPIN14		TRIP Input 14
		0	Trip Input 14 not selected as combinational ORed input
		1	Trip Input 14 selected as combinational ORed input to DCAH mux
12	Reserved		Reserved
11	TRIPIN12		TRIP Input 12
		0	Trip Input 12 not selected as combinational ORed input
		1	Trip Input 12 selected as combinational ORed input to DCAH mux
10	TRIPIN11		TRIP Input 11
		0	Trip Input 11 not selected as combinational ORed input
		1	Trip Input 11 selected as combinational ORed input to DCAH mux
9	TRIPIN10		TRIP Input 10
		0	Trip Input 10 not selected as combinational ORed input
		1	Trip Input 10 selected as combinational ORed input to DCAH mux
8	TRIPIN9		TRIP Input 9
		0	Trip Input 9 not selected as combinational ORed input
		1	Trip Input 9 selected as combinational ORed input to DCAH mux
7	TRIPIN8		TRIP Input 8
		0	Trip Input 8 not selected as combinational ORed input
		1	Trip Input 8 selected as combinational ORed input to DCAH mux
6	TRIPIN7		TRIP Input 7
		0	Trip Input 7 not selected as combinational ORed input
		1	Trip Input 7 selected as combinational ORed input to DCAH mux
5	TRIPIN6		TRIP Input 6
		0	Trip Input 6 not selected as combinational ORed input
		1	Trip Input 6 selected as combinational ORed input to DCAH mux
4	TRIPIN5		TRIP Input 5
		0	Trip Input 5 not selected as combinational ORed input
		1	Trip Input 5 selected as combinational ORed input to DCAH mux
3	TRIPIN4		TRIP Input 4
		0	Trip Input 4 not selected as combinational ORed input
		1	Trip Input 4 selected as combinational ORed input to DCAH mux
2	TRIPIN3		TRIP Input 3
		0	Trip Input 3 not selected as combinational ORed input
		1	Trip Input 3 selected as combinational ORed input to DCAH mux
1	TRIPIN2		TRIP Input 2
		0	Trip Input 2 not selected as combinational ORed input
		1	Trip Input 2 selected as combinational ORed input to DCAH mux
0	TRIPIN1		TRIP Input 1
		0	Trip Input 1 not selected as combinational ORed input
		1	Trip Input 1 selected as combinational ORed input to DCAH mux

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-128. Digital Compare A Low Trip Input Select (DCALTRIPSEL) (EALLOW-protected)

15	14	13	12	11	10	9	8
Reserved	TRIPIN15	TRIPIN14	Reserved	TRIPIN12	TRIPIN11	TRIPIN10	TRIPIN9
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TRIPIN8	TRIPIN7	TRIPIN6	TRIPIN5	TRIPIN4	TRIPIN3	TRIPIN2	TRIPIN1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-78. Digital Compare A Low Trip Input Select (DCALTRIPSEL) Field Descriptions

Bit	Field	Value	Description
15	Reserved		Reserved
14	TRIPIN15		TRIP Input 15
		0	Trip Input 15 not selected as combinational ORed input
		1	Trip Input 15 selected as combinational ORed input to DCAL mux
13	TRIPIN14		TRIP Input 14
		0	Trip Input 14 not selected as combinational ORed input
		1	Trip Input 14 selected as combinational ORed input to DCAL mux
12	Reserved		Reserved
11	TRIPIN12		TRIP Input 12
		0	Trip Input 12 not selected as combinational ORed input
		1	Trip Input 12 selected as combinational ORed input to DCAL mux
10	TRIPIN11		TRIP Input 11
		0	Trip Input 11 not selected as combinational ORed input
		1	Trip Input 11 selected as combinational ORed input to DCAL mux
9	TRIPIN10		TRIP Input 10
		0	Trip Input 10 not selected as combinational ORed input
		1	Trip Input 10 selected as combinational ORed input to DCAL mux
8	TRIPIN9		TRIP Input 9
		0	Trip Input 9 not selected as combinational ORed input
		1	Trip Input 9 selected as combinational ORed input to DCAL mux
7	TRIPIN8		TRIP Input 8
		0	Trip Input 8 not selected as combinational ORed input
		1	Trip Input 8 selected as combinational ORed input to DCAL mux
6	TRIPIN7		TRIP Input 7
		0	Trip Input 7 not selected as combinational ORed input
		1	Trip Input 7 selected as combinational ORed input to DCAL mux
5	TRIPIN6		TRIP Input 6
		0	Trip Input 6 not selected as combinational ORed input
		1	Trip Input 6 selected as combinational ORed input to DCAL mux
4	TRIPIN5		TRIP Input 5
		0	Trip Input 5 not selected as combinational ORed input
		1	Trip Input 5 selected as combinational ORed input to DCAL mux
3	TRIPIN4		TRIP Input 4
		0	Trip Input 4 not selected as combinational ORed input
		1	Trip Input 4 selected as combinational ORed input to DCAL mux
2	TRIPIN3		TRIP Input 3
		0	Trip Input 3 not selected as combinational ORed input
		1	Trip Input 3 selected as combinational ORed input to DCAL mux

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-78. Digital Compare A Low Trip Input Select (DCALTRIPSEL) Field Descriptions (continued)

Bit	Field	Value	Description
1	TRIPIN2		TRIP Input 2
		0	Trip Input 2 not selected as combinational ORed input
		1	Trip Input 2 selected as combinational ORed input to DCAL mux
0	TRIPIN1		TRIP Input 1
		0	Trip Input 1 not selected as combinational ORed input
		1	Trip Input 1 selected as combinational ORed input to DCAL mux

Figure 7-129. Digital Compare B High Trip Input Select (DCBHTRIPSEL) (EALLOW-protected)

15	14	13	12	11	10	9	8
Reserved	TRIPIN15	TRIPIN14	Reserved	TRIPIN12	TRIPIN11	TRIPIN10	TRIPIN9
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TRIPIN8	TRIPIN7	TRIPIN6	TRIPIN5	TRIPIN4	TRIPIN3	TRIPIN2	TRIPIN1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-79. Digital Compare B High Trip Input Select (DCBHTRIPSEL) Field Descriptions

Bit	Field	Value	Description
15	Reserved		Reserved
14	TRIPIN15		TRIP Input 15
		0	Trip Input 15 not selected as combinational ORed input
		1	Trip Input 15 selected as combinational ORed input to DCBH mux
13	TRIPIN14		TRIP Input 14
		0	Trip Input 14 not selected as combinational ORed input
		1	Trip Input 14 selected as combinational ORed input to DCBH mux
12	Reserved		Reserved
11	TRIPIN12		TRIP Input 12
		0	Trip Input 12 not selected as combinational ORed input
		1	Trip Input 12 selected as combinational ORed input to DCBH mux
10	TRIPIN11		TRIP Input 11
		0	Trip Input 11 not selected as combinational ORed input
		1	Trip Input 11 selected as combinational ORed input to DCBH mux
9	TRIPIN10		TRIP Input 10
		0	Trip Input 10 not selected as combinational ORed input
		1	Trip Input 10 selected as combinational ORed input to DCBH mux
8	TRIPIN9		TRIP Input 9
		0	Trip Input 9 not selected as combinational ORed input
		1	Trip Input 9 selected as combinational ORed input to DCBH mux
7	TRIPIN8		TRIP Input 8
		0	Trip Input 8 not selected as combinational ORed input
		1	Trip Input 8 selected as combinational ORed input to DCBH mux
6	TRIPIN7		TRIP Input 7
		0	Trip Input 7 not selected as combinational ORed input
		1	Trip Input 7 selected as combinational ORed input to DCBH mux
5	TRIPIN6		TRIP Input 6
		0	Trip Input 6 not selected as combinational ORed input
		1	Trip Input 6 selected as combinational ORed input to DCBH mux

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-79. Digital Compare B High Trip Input Select (DCBHTRIPSEL) Field Descriptions (continued)

Bit	Field	Value	Description
4	TRIPIN5		TRIP Input 5
		0	Trip Input 5 not selected as combinational ORed input
		1	Trip Input 5 selected as combinational ORed input to DCBH mux
3	TRIPIN4		TRIP Input 4
		0	Trip Input 4 not selected as combinational ORed input
		1	Trip Input 4 selected as combinational ORed input to DCBH mux
2	TRIPIN3		TRIP Input 3
		0	Trip Input 3 not selected as combinational ORed input
		1	Trip Input 3 selected as combinational ORed input to DCBH mux
1	TRIPIN2		TRIP Input 2
		0	Trip Input 2 not selected as combinational ORed input
		1	Trip Input 2 selected as combinational ORed input to DCBH mux
0	TRIPIN1		TRIP Input 1
		0	Trip Input 1 not selected as combinational ORed input
		1	Trip Input 1 selected as combinational ORed input to DCBH mux

Figure 7-130. Digital Compare B Low Trip Input Select (DCBLTRIPSEL) (EALLOW-protected)

15	14	13	12	11	10	9	8
Reserved	TRIPIN15	TRIPIN14	Reserved	TRIPIN12	TRIPIN11	TRIPIN10	TRIPIN9
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TRIPIN8	TRIPIN7	TRIPIN6	TRIPIN5	TRIPIN4	TRIPIN3	TRIPIN2	TRIPIN1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-80. Digital Compare B Low Trip Input Select (DCBLTRIPSEL) Field Descriptions

Bit	Field	Value	Description
15	Reserved		Reserved
14	TRIPIN15		TRIP Input 15
		0	Trip Input 15 not selected as combinational ORed input
		1	Trip Input 15 selected as combinational ORed input to DCBL mux
13	TRIPIN14		TRIP Input 14
		0	Trip Input 14 not selected as combinational ORed input
		1	Trip Input 14 selected as combinational ORed input to DCBL mux
12	Reserved		Reserved
11	TRIPIN12		TRIP Input 12
		0	Trip Input 12 not selected as combinational ORed input
		1	Trip Input 12 selected as combinational ORed input to DCBL mux
10	TRIPIN11		TRIP Input 11
		0	Trip Input 11 not selected as combinational ORed input
		1	Trip Input 11 selected as combinational ORed input to DCBL mux
9	TRIPIN10		TRIP Input 10
		0	Trip Input 10 not selected as combinational ORed input
		1	Trip Input 10 selected as combinational ORed input to DCBL mux
8	TRIPIN9		TRIP Input 9
		0	Trip Input 9 not selected as combinational ORed input
		1	Trip Input 9 selected as combinational ORed input to DCBL mux

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-80. Digital Compare B Low Trip Input Select (DCBLTRIPSEL) Field Descriptions (continued)

Bit	Field	Value	Description
7	TRIPIN8		TRIP Input 8
		0	Trip Input 8 not selected as combinational ORed input
		1	Trip Input 8 selected as combinational ORed input to DCBL mux
6	TRIPIN7		TRIP Input 7
		0	Trip Input 7 not selected as combinational ORed input
		1	Trip Input 7 selected as combinational ORed input to DCBL mux
5	TRIPIN6		TRIP Input 6
		0	Trip Input 6 not selected as combinational ORed input
		1	Trip Input 6 selected as combinational ORed input to DCBL mux
4	TRIPIN5		TRIP Input 5
		0	Trip Input 5 not selected as combinational ORed input
		1	Trip Input 5 selected as combinational ORed input to DCBL mux
3	TRIPIN4		TRIP Input 4
		0	Trip Input 4 not selected as combinational ORed input
		1	Trip Input 4 selected as combinational ORed input to DCBL mux
2	TRIPIN3		TRIP Input 3
		0	Trip Input 3 not selected as combinational ORed input
		1	Trip Input 3 selected as combinational ORed input to DCBL mux
1	TRIPIN2		TRIP Input 2
		0	Trip Input 2 not selected as combinational ORed input
		1	Trip Input 2 selected as combinational ORed input to DCBL mux
0	TRIPIN1		TRIP Input 1
		0	Trip Input 1 not selected as combinational ORed input
		1	Trip Input 1 selected as combinational ORed input to DCBL mux

Copyright © 2012–2019, Texas Instruments Incorporated

7.4.8 GPTRIP Input Select Registers

There are 12 GPTRIP select registers and each has a specific input signal associated with it. Table 7-81 shows the input signal mapping.

Table 7-81. GPTRIP Input Signals

Register	GPTRIP Input Signals
GPTRIP1SEL	$\overline{TZ1}$ and TRIPIN1
GPTRIP2SEL	$\overline{TZ2}$, ADCEXTTRIG, and TRIPIN2
GPTRIP3SEL	$\overline{TZ3}$ and TRIPIN3
GPTRIP4SEL	XINT1 and TRIPIN4
GPTRIP5SEL	XINT2 and TRIPIN5
GPTRIP6SEL	XINT3 and TRIPIN6
GPTRIP7SEL	ECAP1 and TRIPIN7
GPTRIP8SEL	ECAP2 and TRIPIN8
GPTRIP9SEL	ECAP3 and TRIPIN9
GPTRIP10SEL	ECAP4 and TRIPIN10
GPTRIP11SEL	ECAP5 and TRIPIN11
GPTRIP12SEL	ECAP6 and TRIPIN12

Table 7-82. GPIOTRIP Input Select Registers(1)

Register Name	Address	Size (x16)	Description
GPTRIP1SEL	0x5FE0	1	GPTRIP1 (TZ1n) Input Select Register (GPIO0 - GPIO63)
GPTRIP2SEL	0x5FE1	1	GPTRIP2 (TZ2n, ADCEXTTRIG) Input Select Register (GPIO0 - GPIO63)
GPTRIP3SEL	0x5FE2	1	GPTRIP3 (TZ3n) Input Select Register (GPIO0 - GPIO63)
GPTRIP4SEL	0x5FE3	1	GPTRIP4 (XINT1) Input Select Register (GPIO0 - GPIO63)
GPTRIP5SEL	0x5FE4	1	GPTRIP5 (XINT2) Input Select Register (GPIO0 - GPIO63)
GPTRIP6SEL	0x5FE5	1	GPTRIP6 (XINT3) Input Select Register (GPIO0 - GPIO63)
GPTRIP7SEL	0x5FE6	1	GPTRIP7 (ECAP1) Input Select Register (GPIO0 - GPIO63)
GPTRIP8SEL	0x5FE7	1	GPTRIP8 (ECAP2) Input Select Register (GPIO0 - GPIO63)
GPTRIP9SEL	0x5FF0	1	GPTRIP9 (ECAP3) Input Select Register (GPIO0 - GPIO63)
GPTRIP10SEL	0x5FF1	1	GPTRIP10 (ECAP4) Input Select Register (GPIO0 - GPIO63)
GPTRIP11SEL	0x5FF2	1	GPTRIP11 (ECAP5) Input Select Register (GPIO0 - GPIO63)
GPTRIP12SEL	0x5FF3	1	GPTRIP12 (ECAP6) Input Select Register (GPIO0 - GPIO63)

(1) The registers in this table are EALLOW protected.

The GPIO Trip Input Select (GPTRIPxSEL) register is shown in Figure 7-131 and described in Table 7-83.

Figure 7-131. GPIO Trip Input Select Register (GPTRIPxSEL)

15	6	5	0
Reserved		GPTRIPxSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-83. GPIO Trip Input Select Register (GPTRIPxSEL) Field Descriptions

Bit	Field	Value	Description
15-6	Reserved		Reserved
5-0	GPTRIPxSEL		Select GPIO input for GPTRIP signal:
		000000	Select the GPIO0
		000001	Select the GPIO1
		.....	. . .
		111110	Select the GPIO62
		111111	Select the GPIO63

Copyright © 2012–2019, Texas Instruments Incorporated

7.4.9 Event-Trigger Submodule Registers

Figure 7-132 through Figure 7-140 and Table 7-84 through Table 7-92 describe the registers for the event-trigger submodule.

Figure 7-132. Event-Trigger Selection Register (ETSEL)

15	14	12	11	10	8
SOCBEN	SOCBSEL		SOCAEN	SOCASEL	
R/W-0	R/W-0		R/W-0	R/W-0	

7	6	5	4	3	2	0
Reserved	INTSELCMP	SOCBSELCMP	SOCASELCMP	INTEN	INTSEL	
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-84. Event-Trigger Selection Register (ETSEL) Field Descriptions

Bit	Field	Value	Description
15	SOCBEN		Enable the ADC Start of Conversion B (EPWMxSOCB) Pulse
		0	Disable EPWMxSOCB.
		1	Enable EPWMxSOCB pulse.
14-12	SOCBSEL		EPWMxSOCB Selection Options
			These bits determine when a EPWMxSOCB pulse will be generated.
		000	Enable DCBEVT1.soc event
		001	Enable event time-base counter equal to zero. (TBCTR = 0x00)
		010	Enable event time-base counter equal to period (TBCTR = TBPRD)
		011	Enable event time-base counter equal to zero or period (TBCTR = 0x00 or TBCTR = TBPRD). This mode is useful in up-down count mode.
		100	Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing
		101	Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing
		110	Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing
		111	Enable event: time-base counter equal to CMPB when the timer is decrementing or CMPD when the timer is decrementing (*) Event selected is determined by SOCBSELCMP bit.
11	SOCAEN		Enable the ADC Start of Conversion A (EPWMxSOCA) Pulse
		0	Disable EPWMxSOCA.
		1	Enable EPWMxSOCA pulse.
10-8	SOCASEL		EPWMxSOCA Selection Options
			These bits determine when a EPWMxSOCA pulse will be generated.
		000	Enable DCAEVT1.soc event
		001	Enable event time-base counter equal to zero. (TBCTR = 0x00)
		010	Enable event time-base counter equal to period (TBCTR = TBPRD)
		011	Enable event time-base counter equal to zero or period (TBCTR = 0x00 or TBCTR = TBPRD). This mode is useful in up-down count mode.
		100	Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing
		101	Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing
		110	Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing
		111	Enable event: time-base counter equal to CMPB when the timer is decrementing or CMPD when the timer is decrementing (*) Event selected is determined by SOCASELCMP bit.
7	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-84. Event-Trigger Selection Register (ETSEL) Field Descriptions (continued)

Bit	Field	Value	Description
6	INTSELCMP		EPWMxINT Compare Register Selection Options:
		0	Enable event time-base counter equal to CMPA when the timer is incrementing / Enable event time-base counter equal to CMPA when the timer is decrementing / Enable event: time-base counter equal to CMPB when the timer is incrementing / Enable event: time-base counter equal to CMPB when the timer is decrementing to INTSEL selection mux.
		1	Enable event time-base counter equal to CMPC when the timer is incrementing / Enable event time-base counter equal to CMPC when the timer is decrementing / Enable event: time-base counter equal to CMPD when the timer is incrementing / Enable event: time-base counter equal to CMPD when the timer is decrementing to INTSEL selection mux.
5	SOCBSELCMP		EPWMxSOCB Compare Register Selection Options:
		0	Enable event time-base counter equal to CMPA when the timer is incrementing / Enable event time-base counter equal to CMPA when the timer is decrementing / Enable event: time-base counter equal to CMPB when the timer is incrementing / Enable event: time-base counter equal to CMPB when the timer is decrementing to SOCBSEL selection mux.
		1	Enable event time-base counter equal to CMPC when the timer is incrementing / Enable event time-base counter equal to CMPC when the timer is decrementing / Enable event: time-base counter equal to CMPD when the timer is incrementing / Enable event: time-base counter equal to CMPD when the timer is decrementing to SOCBSEL selection mux.
4	SOCASELCMP		EPWMxSOCA Compare Register Selection Options:
		0	Enable event time-base counter equal to CMPA when the timer is incrementing / Enable event time-base counter equal to CMPA when the timer is decrementing / Enable event: time-base counter equal to CMPB when the timer is incrementing / Enable event: time-base counter equal to CMPB when the timer is decrementing to SOCASEL selection mux.
		1	Enable event time-base counter equal to CMPC when the timer is incrementing / Enable event time-base counter equal to CMPC when the timer is decrementing / Enable event: time-base counter equal to CMPD when the timer is incrementing / Enable event: time-base counter equal to CMPD when the timer is decrementing to SOCASEL selection mux.
3	INTEN		Enable ePWM Interrupt (EPWMx_INT) Generation
		0	Disable EPWMx_INT generation
		1	Enable EPWMx_INT generation
2-0	INTSEL		ePWM Interrupt (EPWMx_INT) Selection Options
		000	Reserved
		001	Enable event time-base counter equal to zero. (TBCTR = 0x00)
		010	Enable event time-base counter equal to period (TBCTR = TBPRD)
		011	Enable event time-base counter equal to zero or period (TBCTR = 0x00 or TBCTR = TBPRD). This mode is useful in up-down count mode.
		100	Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing
		101	Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing
		110	Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing
		111	Enable event: time-base counter equal to CMPB when the timer is decrementing or CMPD when the timer is decrementing (*) Event selected is determined by INTSELCMP bit.

Figure 7-133. Event-Trigger Prescale Register (ETPS)

15	14	13	12	11	10	9	8
SOCBCNT		SOCBPRD		SOCACNT		SOCAPRD	
R-0		R/W-0		R-0		R/W-0	
7	6	5	4	3	2	1	0
Reserved		SOCPSSEL	INTPSSEL	INTCNT		INTPRD	
R-0		R/W-0	R/W-0	R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-85. Event-Trigger Prescale Register (ETPS) Field Descriptions

Bit	Field	Value	Description
15-14	SOCBCNT		ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Counter Register
			These bits indicate how many selected ETSEL[SOCBSEL] events have occurred:
		00	No events have occurred.
		01	1 event has occurred.
		10	2 events have occurred.
		11	3 events have occurred.
13-12	SOCBPRD		ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Period Select
			These bits determine how many selected ETSEL[SOCBSEL] events need to occur before an EPWMxSOCB pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCBEN] = 1). The SOCB pulse will be generated even if the status flag is set from a previous start of conversion (ETFLG[SOCB] = 1). Once the SOCB pulse is generated, the ETPS[SOCBCNT] bits will automatically be cleared.
		00	Disable the SOCB event counter. No EPWMxSOCB pulse will be generated
		01	Generate the EPWMxSOCB pulse on the first event: ETPS[SOCBCNT] = 0,1
		10	Generate the EPWMxSOCB pulse on the second event: ETPS[SOCBCNT] = 1,0
		11	Generate the EPWMxSOCB pulse on the third event: ETPS[SOCBCNT] = 1,1
11-10	SOCACNT		ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Counter Register
			These bits indicate how many selected ETSEL[SOCASEL] events have occurred:
		00	No events have occurred.
		01	1 event has occurred.
		10	2 events have occurred.
		11	3 events have occurred.
9-8	SOCAPRD		ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Period Select
			These bits determine how many selected ETSEL[SOCASEL] events need to occur before an EPWMxSOCA pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCAEN] = 1). The SOCA pulse will be generated even if the status flag is set from a previous start of conversion (ETFLG[SOCA] = 1). Once the SOCA pulse is generated, the ETPS[SOCACNT] bits will automatically be cleared.
		00	Disable the SOCA event counter. No EPWMxSOCA pulse will be generated
		01	Generate the EPWMxSOCA pulse on the first event: ETPS[SOCACNT] = 0,1
		10	Generate the EPWMxSOCA pulse on the second event: ETPS[SOCACNT] = 1,0
		11	Generate the EPWMxSOCA pulse on the third event: ETPS[SOCACNT] = 1,1
7-6	Reserved		Reserved
5	SOCPSSEL		EPWMxSOC A/B Pre-Scale Selection Bits
		0	Selects ETPS [SOCACNT/SOCBCNT] and [SOCAPRD/SOCBPRD] registers to determine frequency of events (interrupt once every 0-3 events).
		1	Selects ETSOCPS [SOCACNT2/SOCBCNT2] and [SOCAPRD2/SOCBPRD2] registers to determine frequency of events (interrupt once every 0-15 events).
4	INTPSSEL		EPWMxINTn Pre-Scale Selection Bits
		0	Selects ETPS [INTCNT, and INTPRD] registers to determine frequency of events (interrupt once every 0-3 events).
		1	Selects ETINTPS [INTCNT2, and INTPRD2] registers to determine frequency of events (interrupt once every 0-15 events).
3-2	INTCNT		ePWM Interrupt Event (EPWMx_INT) Counter Register
			These bits indicate how many selected ETSEL[INTSEL] events have occurred. These bits are automatically cleared when an interrupt pulse is generated. If interrupts are disabled, ETSEL[INT] = 0 or the interrupt flag is set, ETFLG[INT] = 1, the counter will stop counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
		00	No events have occurred.
		01	1 event has occurred.
		10	2 events have occurred.
		11	3 events have occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-85. Event-Trigger Prescale Register (ETPS) Field Descriptions (continued)

Bit	Field	Value	Description
1-0	INTPRD		ePWM Interrupt (EPWMx_INT) Period Select These bits determine how many selected ETSEL[INTSEL] events need to occur before an interrupt is generated. To be generated, the interrupt must be enabled (ETSEL[INT] = 1). If the interrupt status flag is set from a previous interrupt (ETFLG[INT] = 1) then no interrupt will be generated until the flag is cleared via the ETCLR[INT] bit. This allows for one interrupt to be pending while another is still being serviced. Once the interrupt is generated, the ETPS[INTCNT] bits will automatically be cleared. Writing to the INTPRD bits in different situations will cause the following results: • Writing an INTPRD value that is GREATER or equal to the current counter value will reset the INTCNT = 0 • Writing an INTPRD value that is equal to the current counter value will trigger an interrupt if it is enabled and the status flag is cleared (and INTCNT will also be cleared to 0) • Writing an INTPRD value that is LESS than the current counter value will result in undefined behavior (that is, INTCNT stops counting because INTPRD is below INTCNT, and interrupt will never fire). If a counter event occurs at the same instant as a new zero or non-zero INTPRD value is written, the counter is incremented.
		00	Disable the interrupt event counter. No interrupt will be generated and ETFRC[INT] is ignored.
		01	Generate an interrupt on the first event INTCNT = 01 (first event)
		10	Generate interrupt on ETPS[INTCNT] = 1,0 (second event)
		11	Generate interrupt on ETPS[INTCNT] = 1,1 (third event)

Figure 7-134. Event-Trigger Interrupt Pre-Scale Register (ETINTPS)

15	8
Reserved	
R-0	

7	4	3	0
INTCNT2		INTPRD2	
R-0:0		R/W-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-86. Event-Trigger Interrupt Pre-Scale Register (ETINTPS) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved		Reserved
7-4	INTCNT2		$\overline{\text{EPWMxINT}}$ Counter 2 When ETPS[INTPSSEL]=1, these bits indicate how many selected events have occurred:
		0000	No events
		0001	1 event
		0010	2 events
		0011	3 events
		0100	4 events
		. . .	. . .
		1111	15 events

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-86. Event-Trigger Interrupt Pre-Scale Register (ETINTPS) Field Descriptions (continued)

Bit	Field	Value	Description
3-0	INTPRD2		$\overline{\text{EPWMxINT}}$ Period 2 Select
			When ETPS[INTPSSEL] = 1, these bits select how many selected events need to occur before an interrupt is generated:
		0000	Disable counter
		0001	Generate interrupt on INTCNT = 1 (first event)
		0010	Generate interrupt on INTCNT = 2 (second event)
		0011	Generate interrupt on INTCNT = 3 (third event)
		0100	Generate interrupt on INTCNT = 4 (fourth event)
		. . .	. . .
		1111	Generate interrupt on INTCNT = 15 (fifteenth event)

Figure 7-135. Event-Trigger SOC Pre-Scale Register (ETSOCPS)

15	12	11	8
SOCBCNT2		SOCBPRD2	
R-0:0		R/W-0:0	

7	4	3	0
SOCACNT2		SOCAPRD2	
R-0:0		R/W-0:0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-87. Event-Trigger SOC Pre-Scale Register (ETSOCPS) Field Descriptions

Bit	Field	Value	Description
15-12	SOCBCNT2		EPWMxSOCB Counter 2
			When ETPS[SOCPSSEL] = 1, these bits indicate how many selected events have occurred:
		0000	No events
		0001	1 event
		0010	2 events
		0011	3 events
		0100	4 events
		. . .	. . .
		1111	15 events
11-8	SOCBPRD2		EPWMxSOCB Period 2 Select
			When ETPS[SOCPSSEL] = 1, these bits select how many selected event need to occur before an SOCB pulse is generated:
		0000	Disable counter
		0001	Generate interrupt on SOCBCNT2 = 1 (first event)
		0010	Generate interrupt on SOCBCNT2 = 2 (second event)
		0011	Generate interrupt on SOCBCNT2 = 3 (third event)
		0100	Generate interrupt on SOCBCNT2 = 4 (fourth event)
		. . .	. . .
		1111	Generate interrupt on SOCBCNT2 = 15 (fifteenth event)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-87. Event-Trigger SOC Pre-Scale Register (ETSOCPS) Field Descriptions (continued)

Bit	Field	Value	Description
7-4	SOCACNT2		EPWMxSOCA Counter 2
			When ETPS[SOCPSSEL] = 1, these bits indicate how many selected events have occurred:
		0000	No events
		0001	1 event
		0010	2 events
		0011	3 events
		0100	4 events
		. . .	. . .
		1111	15 events
3-0	SOCAPRD2		EPWMxSOCA Period 2 Select
			When ETPS[SOCPSSEL] = 1, these bits select how many selected event need to occur before an SOCA pulse is generated:
		0000	Disable counter
		0001	Generate interrupt on SOCACNT2 = 1 (first event)
		0010	Generate interrupt on SOCACNT2 = 2 (second event)
		0011	Generate interrupt on SOCACNT2 = 3 (third event)
		0100	Generate interrupt on SOCACNT2 = 4 (fourth event)
		. . .	. . .
		1111	Generate interrupt on SOCACNT2 = 15 (fifteenth event)

Figure 7-136. Event-Trigger Flag Register (ETFLG)

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		SOCB	SOCA	Reserved	INT
R-0		R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-88. Event-Trigger Flag Register (ETFLG) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3	SOCB		Latched ePWM ADC Start-of-Conversion B (EPWMxSOCB) Status Flag
		0	Indicates no EPWMxSOCB event occurred
		1	Indicates that a start of conversion pulse was generated on EPWMxSOCB. The EPWMxSOCB output will continue to be generated even if the flag bit is set.
2	SOCA		Latched ePWM ADC Start-of-Conversion A (EPWMxSOCA) Status Flag
			Unlike the ETFLG[INT] flag, the EPWMxSOCA output will continue to pulse even if the flag bit is set.
		0	Indicates no event occurred
		1	Indicates that a start of conversion pulse was generated on EPWMxSOCA. The EPWMxSOCA output will continue to be generated even if the flag bit is set.
1	Reserved		Reserved
0	INT		Latched ePWM Interrupt (EPWMx_INT) Status Flag
		0	Indicates no event occurred
		1	Indicates that an ePWMx interrupt (EWPMx_INT) was generated. No further interrupts will be generated until the flag bit is cleared. Up to one interrupt can be pending while the ETFLG[INT] bit is still set. If an interrupt is pending, it will not be generated until after the ETFLG[INT] bit is cleared. Refer to Figure 7-44.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-137. Event-Trigger Clear Register and Mirror Register (ETCLR / ETCLRM)

15					8
Reserved					
R = 0					
7	4	3	2	1	0
Reserved		SOCB	SOCA	Reserved	INT
R-0		R/W-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-89. Event-Trigger Clear Register and Mirror Register (ETCLR / ETCLRM) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3	SOCB		ePWM ADC Start-of-Conversion B (EPWMxSOCB) Flag Clear Bit
		0	Writing a 0 has no effect. Always reads back a 0
		1	Clears the ETFLG[SOCB] flag bit
2	SOCA		ePWM ADC Start-of-Conversion A (EPWMxSOCA) Flag Clear Bit
		0	Writing a 0 has no effect. Always reads back a 0
		1	Clears the ETFLG[SOCA] flag bit
1	Reserved		Reserved
0	INT		ePWM Interrupt (EPWMx_INT) Flag Clear Bit
		0	Writing a 0 has no effect. Always reads back a 0
		1	Clears the ETFLG[INT] flag bit and enable further interrupts pulses to be generated

Figure 7-138. Event-Trigger Force Register (ETFRC)

15					8
Reserved					
R-0					
7	4	3	2	1	0
Reserved		SOCB	SOCA	Reserved	INT
R-0		R/W-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-90. Event-Trigger Force Register (ETFRC) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3	SOCB		SOCB Force Bit. The SOCB pulse will only be generated if the event is enabled in the ETSEL register. The ETFLG[SOCB] flag bit will be set regardless.
		0	Has no effect. Always reads back a 0.
		1	Generates a pulse on EPWMxSOCB and sets the SOCBFLG bit. This bit is used for test purposes.
2	SOCA		SOCA Force Bit. The SOCA pulse will only be generated if the event is enabled in the ETSEL register. The ETFLG[SOCA] flag bit will be set regardless.
		0	Writing 0 to this bit will be ignored. Always reads back a 0.
		1	Generates a pulse on EPWMxSOCA and set the SOCAFLG bit. This bit is used for test purposes.
1	Reserved	0	Reserved
0	INT		INT Force Bit. The interrupt will only be generated if the event is enabled in the ETSEL register. The INT flag bit will be set regardless.
		0	Writing 0 to this bit will be ignored. Always reads back a 0.
		1	Generates an interrupt on $\overline{\text{EPWMxINT}}$ and set the INT flag bit. This bit is used for test purposes.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 7-139. Event-Trigger Counter Initialization Control Register (ETCNTINITCTL)

15	14	13	12	11	10	9	8
SOCBINITEN	SOCAINITEN	INTINITEN	SOCBINITFRC	SOCAINITFRC	INTINITFRC	Reserved	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0:00	

7	0
Reserved	
R-0:00	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7-91. Event-Trigger Counter Initialization Control Register (ETCNTINITCTL) Field Descriptions

Bit	Field	Value	Description
15	SOCBINITEN		EPWMxSOCB Counter 2 Initialization Enable
		0	Has no effect.
		1	Enable initialization of EPWMxSOCB counter with contents of ETCNTINIT[SOCBINIT] on a SYNC event or software force.
14	SOCAINITEN		EPWMxSOCA Counter 2 Initialization Enable
		0	Has no effect.
		1	Enable initialization of EPWMxSOCA counter with contents of ETCNTINIT[SOCAINIT] on a SYNC event or software force.
13	INTINITEN		EPWMxINT Counter 2 Initialization Enable
		0	Has no effect.
		1	Enable initialization of EPWMxINT counter 2 with contents of ETCNTINIT[INTINIT] on a SYNC event or software force.
12	SOCBINITFRC		EPWMxSOCB Counter 2 Initialization Force
		0	Has no effect.
		1	This bit forces the ET EPWMxSOCB counter to be initialized with the contents of ETCNTINIT[SOCBINIT].
11	SOCAINITFRC		EPWMxSOCA Counter 2 Initialization Force
		0	Has no effect.
		1	This bit forces the ET EPWMxSOCA counter to be initialized with the contents of ETCNTINIT[SOCAINIT].
10	INTINITFRC		EPWMxINT Counter 2 Initialization Force
		0	Has no effect.
		1	This bit forces the ET EPWMxINT counter to be initialized with the contents of ETCNTINIT[INTINIT].
9-0	Reserved		Reserved

Figure 7-140. Event-Trigger Counter Initialization Register (ETCNTINIT)

15	12	11	8
Reserved		SOCBINIT	
R-0:00		R/W-0:00	

7	4	3	0
SOCAINIT		INTINIT	
R/W-0:00		R/W-0:00	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 7-92. Event-Trigger Counter Initialization Register (ETCNTINIT) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved		Reserved
11-8	SOCBINIT	0000-1111	EPWMxSOCB Counter 2 Initialization Bits The ET EPWMxSOCB counter is initialized with the contents of this register on an ePWM SYNC event or a software force.
7-4	SOCAINIT	0000-1111	EPWMxSOCA Counter 2 Initialization Bits The ET EPWMxSOCA counter is initialized with the contents of this register on an ePWM SYNC event or a software force.
3-0	INTINIT	0000-1111	EPWMxINT Counter 2 Initialization Bits The ET EPWMxINT counter is initialized with the contents of this register on an ePWM SYNC event or a software force.

Copyright © 2012–2019, Texas Instruments Incorporated

7.4.10 Proper Interrupt Initialization Procedure

When the ePWM peripheral clock is enabled it may be possible that interrupt flags may be set due to spurious events due to the ePWM registers not being properly initialized. The proper procedure for initializing the ePWM peripheral is as follows:

1. Disable global interrupts (CPU INTM flag)
2. Disable ePWM interrupts
3. Set TBCLKSYNC=0
4. Initialize peripheral registers
5. Set TBCLKSYNC=1
6. Clear any spurious ePWM flags (including PIEIFR)
7. Enable ePWM interrupts
8. Enable global interrupts

Copyright © 2012–2019, Texas Instruments Incorporated

C28 Enhanced Capture (eCAP) Module

This chapter describes the enhanced capture (eCAP) module, which is used in systems where accurate timing of external events is important.

The eCAP module described in this reference guide is a Type 0 eCAP. See the *TMS320C28xx, 28xxx DSP Peripheral Reference Guide* (SPRU566) for a list of all devices with a eCAP module of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

Topic **Page**

8.1 Introduction **791**
8.2 Description **791**
8.3 Capture and APWM Operating Mode **792**
8.4 Capture Mode Description **793**
8.5 Capture Module - Control and Status Registers **801**
8.6 Register Mapping **809**
8.7 Application of the ECAP Module **809**
8.8 Application of the APWM Mode **818**

Copyright © 2012–2019, Texas Instruments Incorporated

8.1 Introduction

Features for eCAP include:

- Speed measurements of rotating machinery (for example, toothed sprockets sensed via Hall sensors)
- Elapsed time measurements between position sensor pulses
- Period and duty cycle measurements of pulse train signals
- Decoding current or voltage amplitude derived from duty cycle encoded current/voltage sensors

The eCAP module described in this guide includes the following features:

- 4-event time-stamp registers (each 32 bits)
- Edge polarity selection for up to four sequenced time-stamp capture events
- Interrupt on either of the four events
- Single shot capture of up to four event time-stamps
- Continuous mode capture of time-stamps in a four-deep circular buffer
- Absolute time-stamp capture
- Difference (Delta) mode time-stamp capture
- All above resources dedicated to a single input pin
- When not used in capture mode, the ECAP module can be configured as a single channel PWM output

8.2 Description

The eCAP module represents one complete capture channel that can be instantiated multiple times depending on the target device. In the context of this guide, one eCAP channel has the following independent key resources:

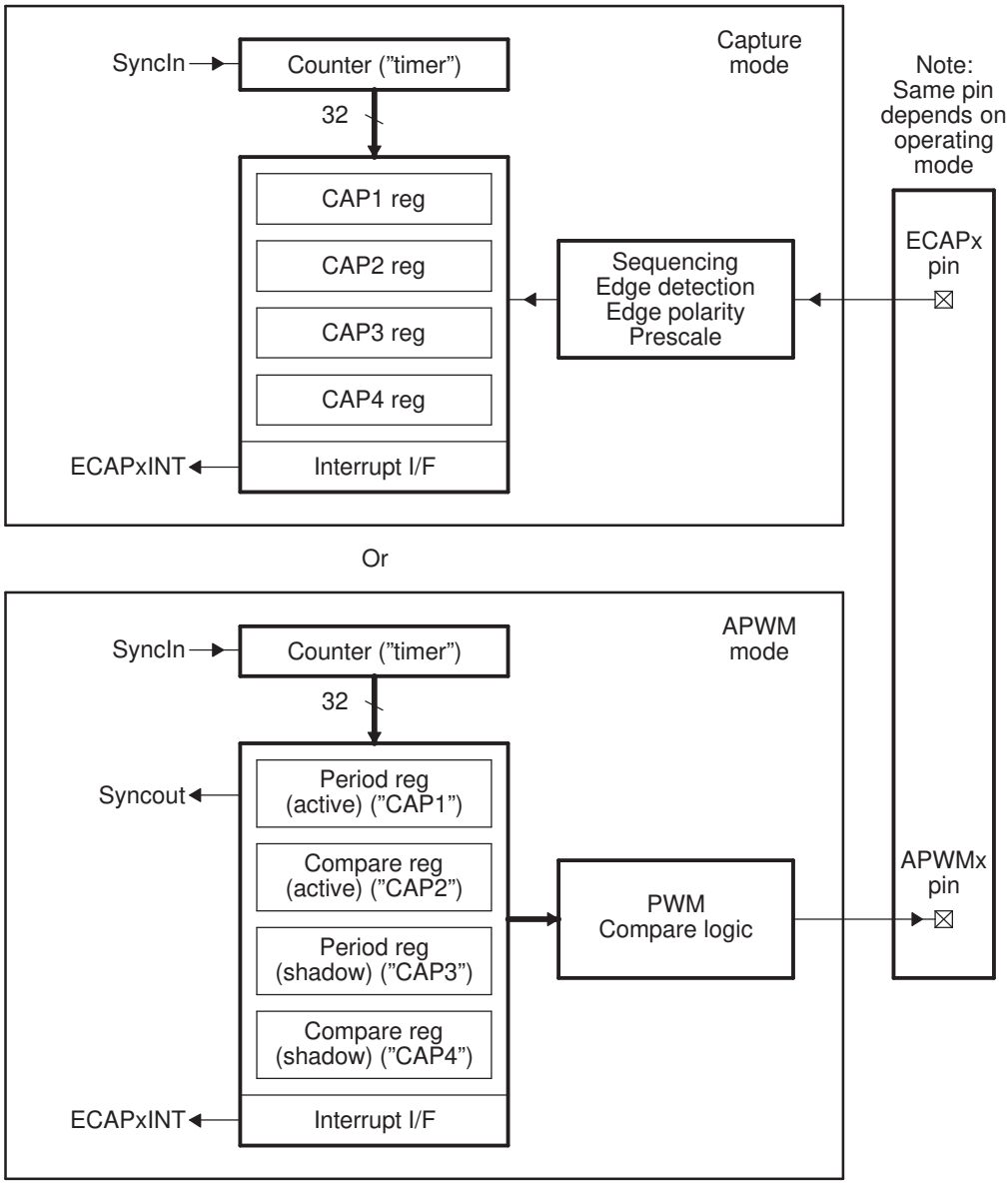
- Dedicated input capture pin
- 32-bit time base (counter)
- 4 x 32-bit time-stamp capture registers (CAP1-CAP4)
- 4-stage sequencer (Modulo4 counter) that is synchronized to external events, ECAP pin rising/falling edges.
- Independent edge polarity (rising/falling edge) selection for all 4 events
- Input capture signal prescaling (from 2-62)
- One-shot compare register (2 bits) to freeze captures after 1 to 4 time-stamp events
- Control for continuous time-stamp captures using a 4-deep circular buffer (CAP1-CAP4) scheme
- Interrupt capabilities on any of the 4 capture events

Copyright © 2012–2019, Texas Instruments Incorporated

8.3 Capture and APWM Operating Mode

You can use the eCAP module resources to implement a single-channel PWM generator (with 32 bit capabilities) when it is not being used for input captures. The counter operates in count-up mode, providing a time-base for asymmetrical pulse width modulation (PWM) waveforms. The CAP1 and CAP2 registers become the active period and compare registers, respectively, while CAP3 and CAP4 registers become the period and capture shadow registers, respectively. Figure 8-1 is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

Figure 8-1. Capture and APWM Modes of Operation



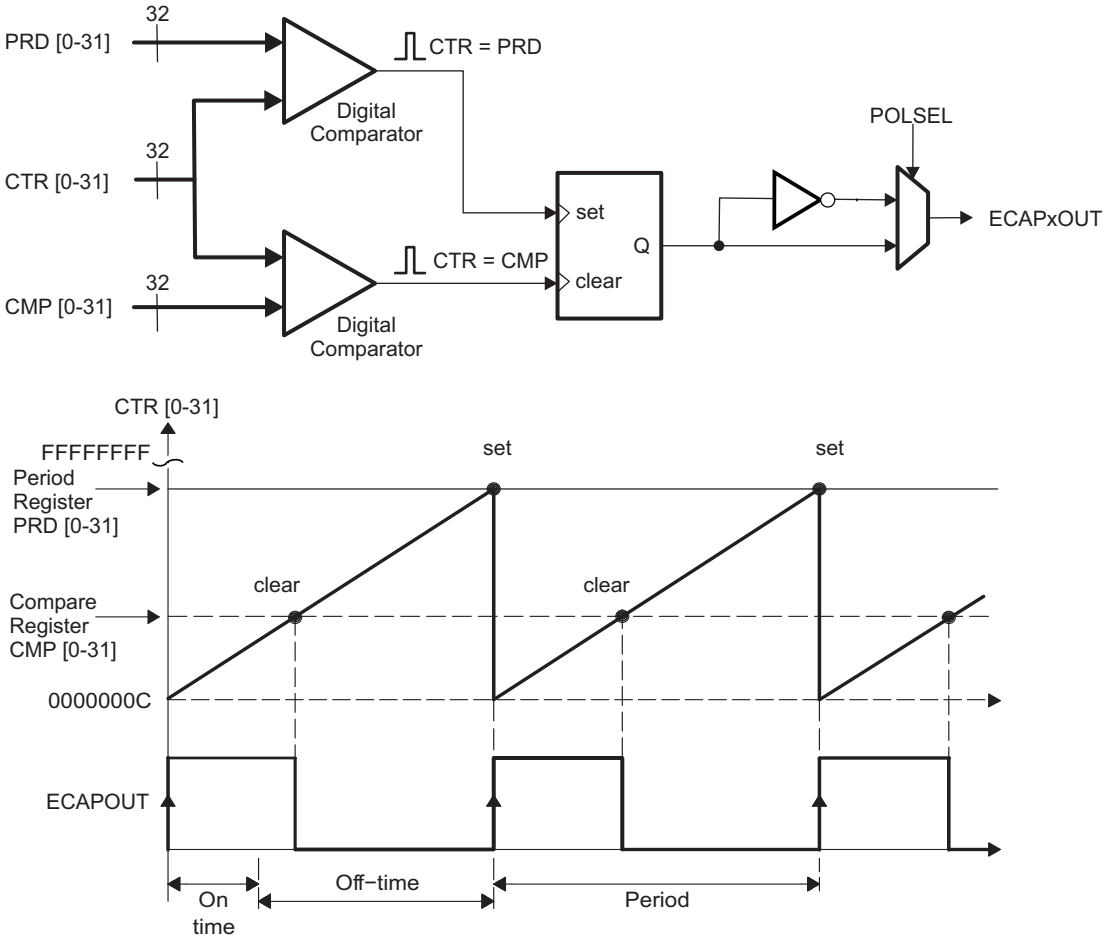
A A single pin is shared between CAP and APWM functions. In capture mode, it is an input; in APWM mode, it is an output.

B In APWM mode, writing any value to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

Figure 8-2 further descries the output of the eCAP in APWM mode based on the CMP and PRD values.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 8-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode

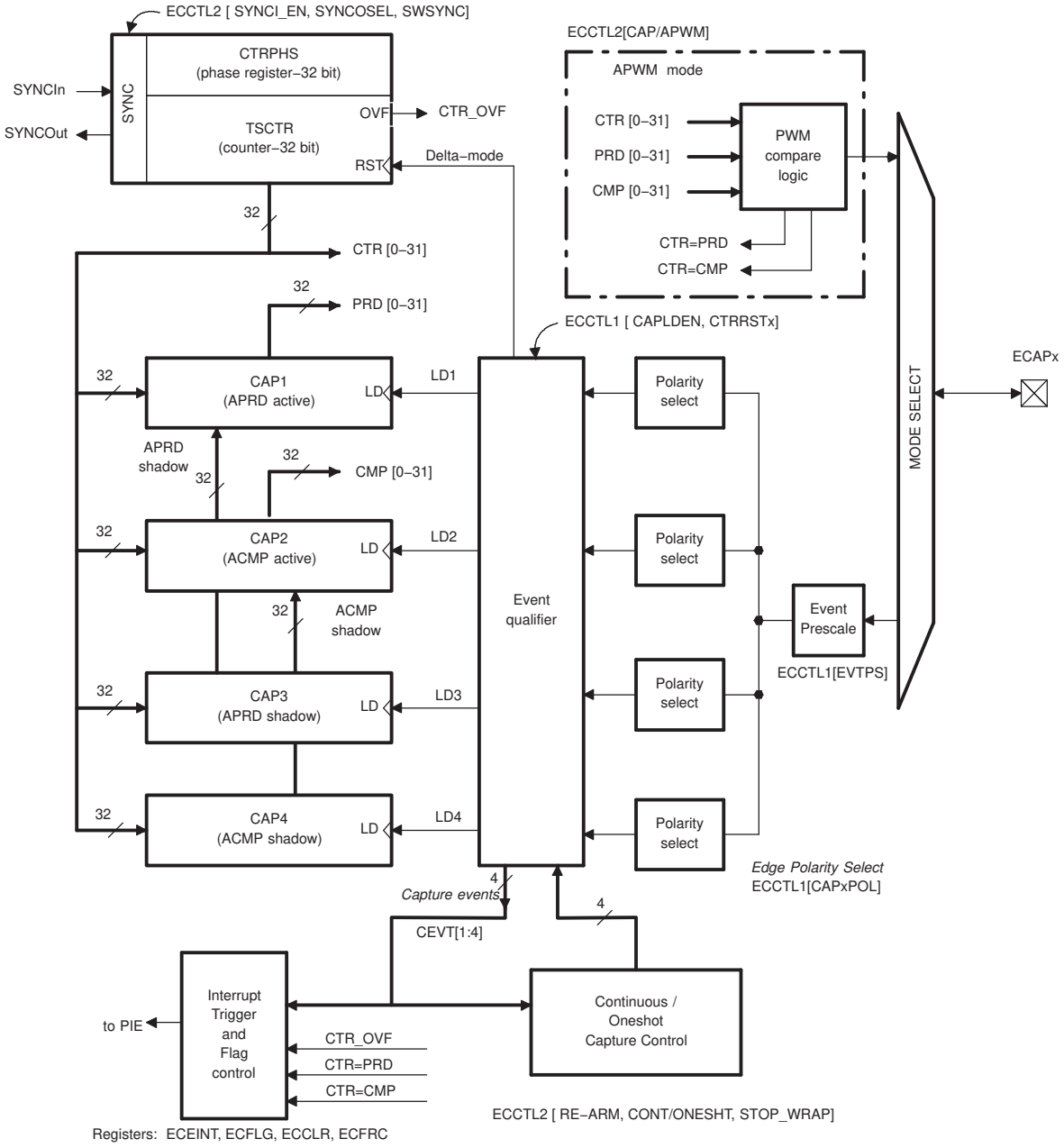


8.4 Capture Mode Description

Figure 8-3 shows the various components that implement the capture function.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 8-3. Capture Function Diagram



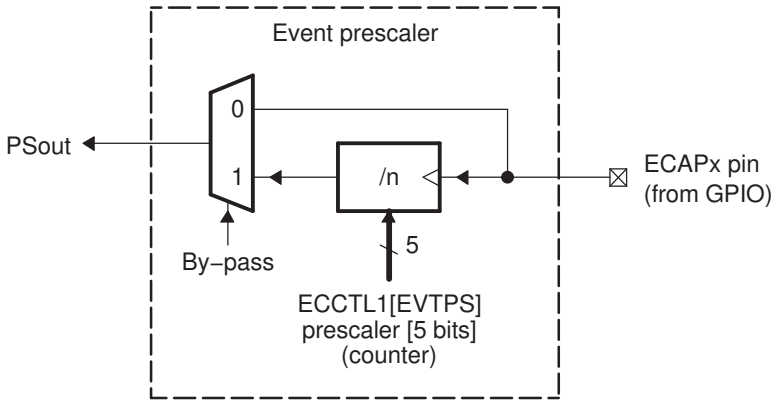
8.4.1 Event Prescaler

- An input capture signal (pulse train) can be prescaled by N = 2-62 (in multiples of 2) or can bypass the prescaler.

  This is useful when very high frequency signals are used as inputs. Figure 8-4 shows a functional diagram and Figure 8-5 shows the operation of the prescale function.

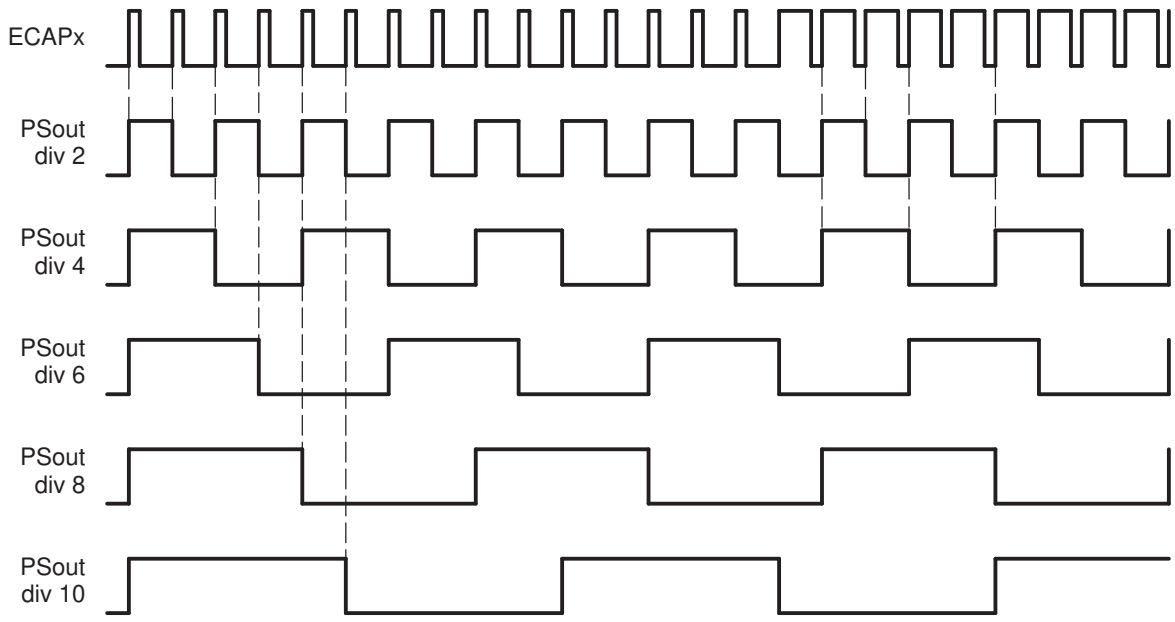
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 8-4. Event Prescale Control



A When a prescale value of 1 is chosen (ECCTL1[13:9] = 0,0,0,0,0) the input capture signal by-passes the prescale logic completely.

Figure 8-5. Prescale Function Waveforms



8.4.2 Edge Polarity Select and Qualifier

- Four independent edge polarity (rising edge/falling edge) selection MUXes are used, one for each capture event.
- Each edge (up to 4) is event qualified by the Modulo4 sequencer.
- The edge event is gated to its respective CAPx register by the Mod4 counter. The CAPx register is loaded on the falling edge.

8.4.3 Continuous/One-Shot Control

- The Mod4 (2 bit) counter is incremented via edge qualified events (CEVT1-CEVT4).
- The Mod4 counter continues counting (0->1->2->3->0) and wraps around unless stopped.
- A 2-bit stop register is used to compare the Mod4 counter output, and when equal stops the Mod4 counter and inhibits further loads of the CAP1-CAP4 registers. This occurs during one-shot operation.

Copyright © 2012–2019, Texas Instruments Incorporated

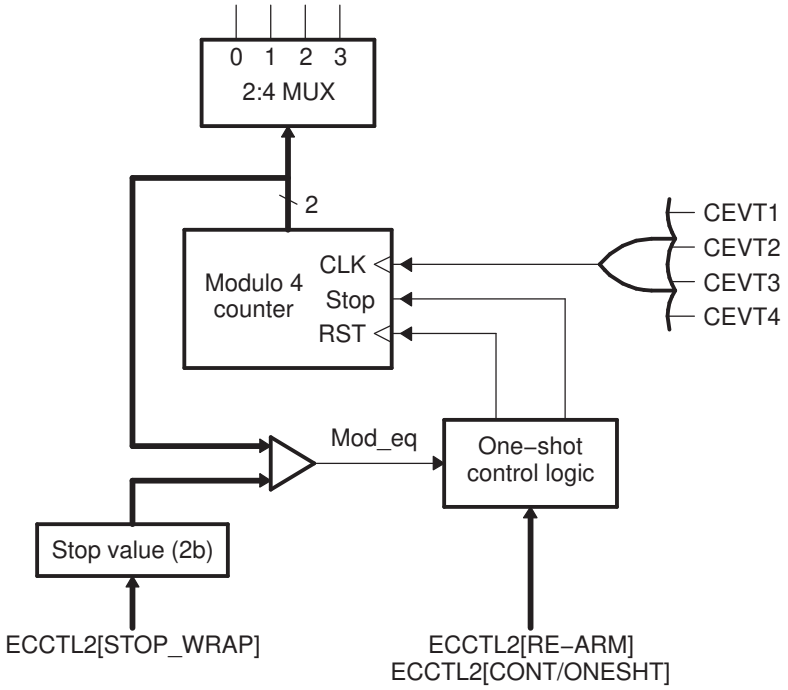
The continuous/one-shot block controls the start/stop and reset (zero) functions of the Mod4 counter via a mono-shot type of action that can be triggered by the stop-value comparator and re-armed via software control.

Once armed, the eCAP module waits for 1-4 (defined by stop-value) capture events before freezing both the Mod4 counter and contents of CAP1-4 registers (time-stamps).

Re-arming prepares the eCAP module for another capture sequence. Also re-arming clears (to zero) the Mod4 counter and permits loading of CAP1-4 registers again, providing the CAPLDEN bit is set.

In continuous mode, the Mod4 counter continues to run (0->1->2->3->0, the one-shot action is ignored, and capture values continue to be written to CAP1-4 in a circular buffer sequence.

Figure 8-6. Details of the Continuous/One-shot Block



8.4.4 32-Bit Counter and Phase Control

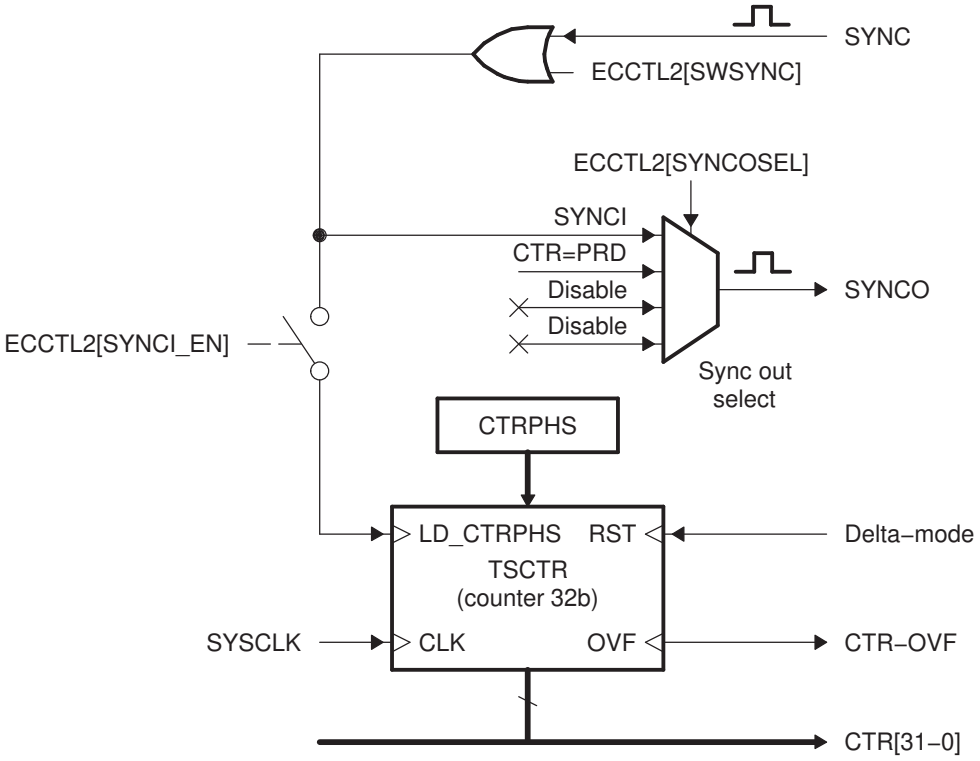
This counter provides the time-base for event captures, and is clocked via the system clock.

A phase register is provided to achieve synchronization with other counters, via a hardware and software forced sync. This is useful in APWM mode when a phase offset between modules is needed.

On any of the four event loads, an option to reset the 32-bit counter is given. This is useful for time difference capture. The 32-bit counter value is captured first, then it is reset to 0 by any of the LD1-LD4 signals.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 8-7. Details of the Counter and Synchronization Block



8.4.5 CAP1-CAP4 Registers

These 32-bit registers are fed by the 32-bit counter timer bus, CTR[0-31] and are loaded (capture a time-stamp) when their respective LD inputs are strobed.

Loading of the capture registers can be inhibited via control bit CAPLDEN. During one-shot operation, this bit is cleared (loading is inhibited) automatically when a stop condition occurs, StopValue = Mod4.

CAP1 and CAP2 registers become the active period and compare registers, respectively, in APWM mode.

CAP3 and CAP4 registers become the respective shadow registers (APRD and ACMP) for CAP1 and CAP2 during APWM operation.

8.4.6 Interrupt Control

An Interrupt can be generated on capture events (CEVT1-CEVT4, CTROVF) or APWM events (CTR = PRD, CTR = CMP).

A counter overflow event (FFFFFFFF->00000000) is also provided as an interrupt source (CTROVF).

The capture events are edge and sequencer qualified (ordered in time) by the polarity select and Mod4 gating, respectively.

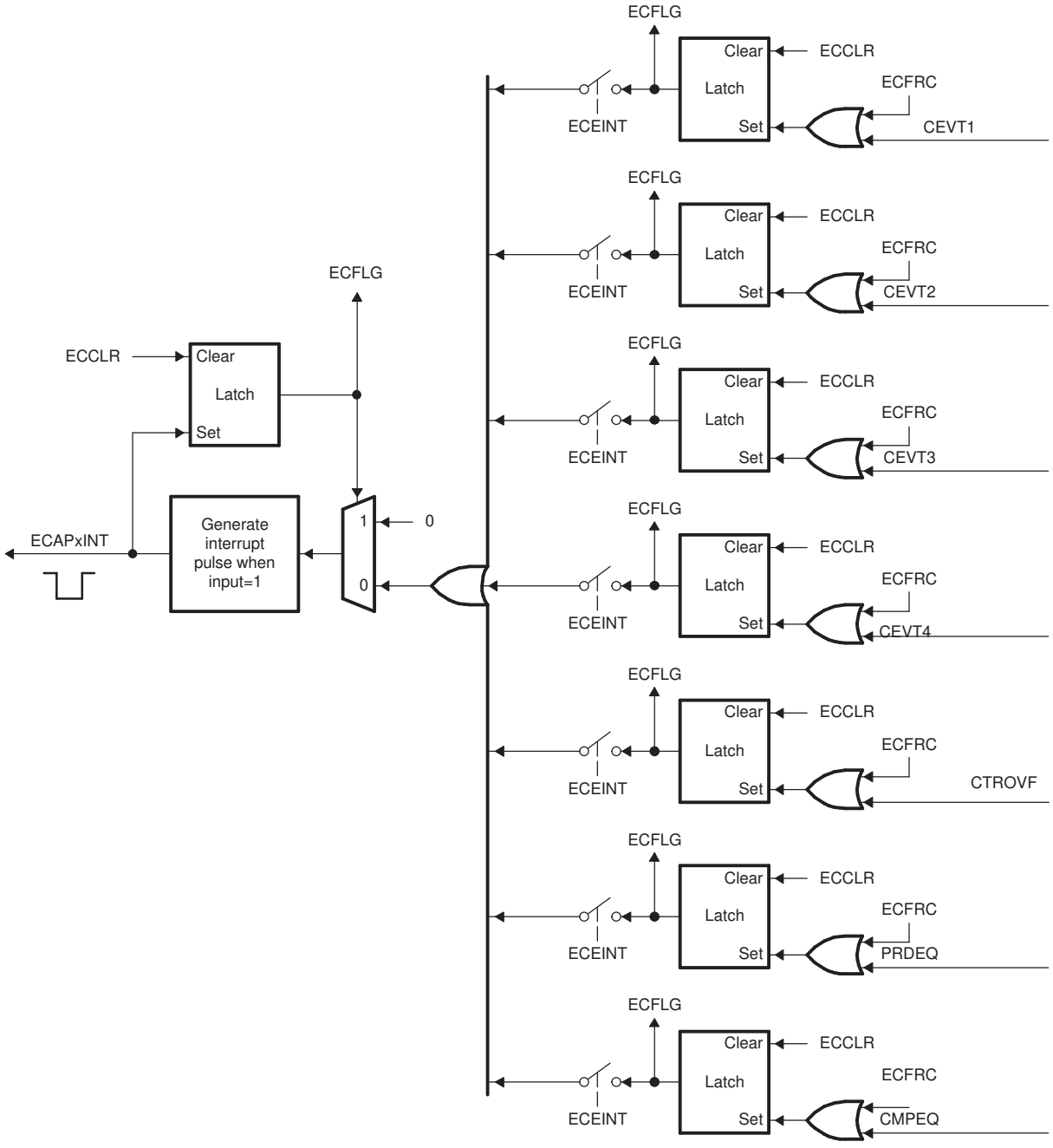
One of these events can be selected as the interrupt source (from the eCAPx module) going to the PIE.

Seven interrupt events (CEVT1, CEVT2, CEVT3, CEVT4, CNTOVF, CTR=PRD, CTR=CMP) can be generated. The interrupt enable register (ECEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (ECFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated to the PIE only if any of the interrupt events are enabled, the flag bit is 1, and the INT flag bit is 0. The interrupt service routine must clear the global interrupt flag bit and the serviced event via the interrupt clear register (ECCLR) before any other interrupt pulses are generated. You can force an interrupt event via the interrupt force register (ECFRC). This is useful for test purposes.

Copyright © 2012–2019, Texas Instruments Incorporated

Note: The CEVT1, CEVT2, CEVT3, CEVT4 flags are only active in capture mode (ECCTL2[CAP/APWM == 0]). The CTR=PRD, CTR=CMP flags are only valid in APWM mode (ECCTL2[CAP/APWM == 1]). CNTOVF flag is valid in both modes.

Figure 8-8. Interrupts in eCAP Module



8.4.7 Shadow Load and Lockout Control

In capture mode, this logic inhibits (locks out) any shadow loading of CAP1 or CAP2 from APRD and ACMP registers, respectively.

In APWM mode, shadow loading is active and two choices are permitted:

- Immediate - APRD or ACMP are transferred to CAP1 or CAP2 immediately upon writing a new value.
- On period equal, CTR[31:0] = PRD[31:0]

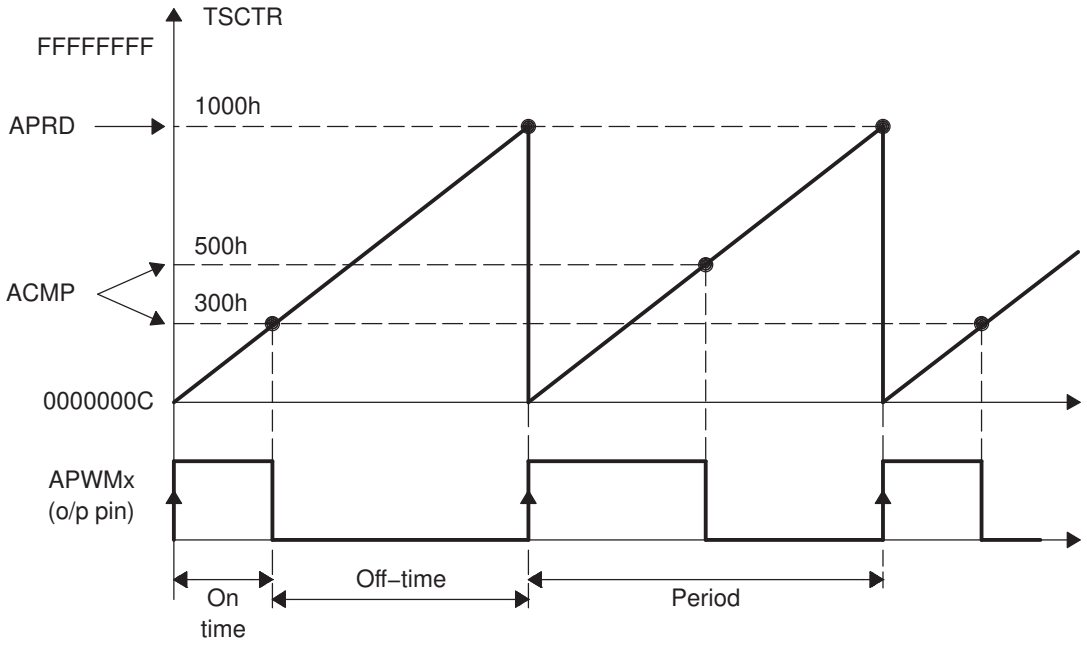
Copyright © 2012–2019, Texas Instruments Incorporated

8.4.8 APWM Mode Operation

Main operating highlights of the APWM section:

- The time-stamp counter bus is made available for comparison via 2 digital (32-bit) comparators.
- When CAP1/2 registers are not used in capture mode, their contents can be used as Period and Compare values in APWM mode.
- Double buffering is achieved via shadow registers APRD and ACMP (CAP3/4). The shadow register contents are transferred over to CAP1/2 registers either immediately upon a write, or on a CTR = PRD trigger.
- In APWM mode, writing to CAP1/CAP2 active registers will also write the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 will invoke the shadow mode.
- During initialization, you must write to the active registers for both period and compare. This automatically copies the initial values into the shadow values. For subsequent compare updates, during run-time, you only need to use the shadow registers.

Figure 8-9. PWM Waveform Details Of APWM Mode Operation



The behavior of APWM active high mode (APWMPOL == 0) is as follows:

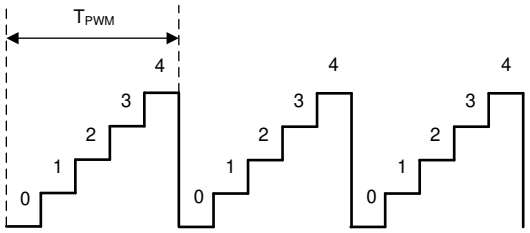
```
CMP = 0x00000000, output low for duration of period (0% duty)
CMP = 0x00000001, output high 1 cycle
CMP = 0x00000002, output high 2 cycles
CMP = PERIOD, output high except for 1 cycle (<100% duty)
CMP = PERIOD+1, output high for complete period (100% duty)
CMP > PERIOD+1, output high for complete period
```

The behavior of APWM active low mode (APWMPOL == 1) is as follows:

```
CMP = 0x00000000, output high for duration of period (0% duty)
CMP = 0x00000001, output low 1 cycle
CMP = 0x00000002, output low 2 cycles
CMP = PERIOD, output low except for 1 cycle (<100% duty)
CMP = PERIOD+1, output low for complete period (100% duty)
CMP > PERIOD+1, output low for complete period
```


Copyright © 2012–2019, Texas Instruments Incorporated

Figure 8-10. Time-Base Frequency and Period Calculation



$$T_{PWM} = (CAP1 + 1) \times T_{TSCTR}$$

$$F_{PWM} = \frac{1}{T_{PWM}}$$

Copyright © 2012–2019, Texas Instruments Incorporated

8.5 Capture Module - Control and Status Registers

Figure 8-11. Time-Stamp Counter Register (TSCTR)

31	0
TSCTR	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-1. Time-Stamp Counter Register (TSCTR) Field Descriptions

Bit(s)	Field	Description
31:0	TSCTR	Active 32-bit counter register that is used as the capture time-base

Figure 8-12. Counter Phase Control Register (CTRPHS)

31	0
CTRPHS	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-2. Counter Phase Control Register (CTRPHS) Field Descriptions

Bit(s)	Field	Description
31:0	CTRPHS	Counter phase value register that can be programmed for phase lag/lead. This register shadows TSCTR and is loaded into TSCTR upon either a SYNCI event or S/W force via a control bit. Used to achieve phase control synchronization with respect to other eCAP and EPWM time-bases.

Figure 8-13. Capture-1 Register (CAP1)

31	0
CAP1	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-3. Capture-1 Register (CAP1) Field Descriptions

Bit(s)	Field	Description
31:0	CAP1	This register can be loaded (written) by :) Time-Stamp (counter value TSCTR) during a capture event) Software - may be useful for test purposes / initialization) APRD shadow register (CAP3) when used in APWM mode

Figure 8-14. Capture-2 Register (CAP2)

31	0
CAP2	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-4. Capture-2 Register (CAP2) Field Descriptions

Bit(s)	Field	Description
31:0	CAP2	This register can be loaded (written) by: • Time-Stamp (counter value) during a capture event • Software - may be useful for test purposes • APRD shadow register (CAP4) when used in APWM mode

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: In APWM mode, writing to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

Figure 8-15. Capture-3 Register (CAP3)

31	0
CAP3	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-5. Capture-3 Register (CAP3) Field Descriptions

Bit(s)	Field	Description
31:0	CAP3	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the period shadow (APRD) register. You update the PWM period value through this register. In this mode, CAP3 (APRD) shadows CAP1.

Figure 8-16. Capture-4 Register (CAP4)

31	0
CAP4	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-6. Capture-4 Register (CAP4) Field Descriptions

Bit(s)	Field	Description
31:0	CAP4	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the compare shadow (ACMP) register. You update the PWM compare value via this register. In this mode, CAP4 (ACMP) shadows CAP2.

Figure 8-17. ECAP Control Register 1 (ECCTL1)

15	14	13	12	11	10	9	8
FREE/SOFT		PRESCALE					CAPLDEN
R/W-0		R/W-0					R/W-0
7	6	5	4	3	2	1	0
CTRRST4	CAP4POL	CTRRST3	CAP3POL	CTRRST2	CAP2POL	CTRRST1	CAP1POL
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-7. ECAP Control Register 1 (ECCTL1) Field Descriptions

Bit(s)	Field	Value	Description
15:14	FREE/SOFT		Emulation Control
		00	TSCTR counter stops immediately on emulation suspend
		01	TSCTR counter runs until = 0
		1x	TSCTR counter is unaffected by emulation suspend (Run Free)
13:9	PRESCALE		Event Filter prescale select
		00000	Divide by 1 (i.e,. no prescale, by-pass the prescaler)
		00001	Divide by 2
		00010	Divide by 4
		00011	Divide by 6

Copyright © 2012–2019, Texas Instruments Incorporated

Table 8-7. ECAP Control Register 1 (ECCTL1) Field Descriptions (continued)

Bit(s)	Field	Value	Description
		00100	Divide by 8
		00101	Divide by 10
		...	
		11110	Divide by 60
		11111	Divide by 62
8	CAPLDEN		Enable Loading of CAP1-4 registers on a capture event. Note that this bit does not disable CEVTn events from being generated.
		0	Disable CAP1-4 register loads at capture event time.
		1	Enable CAP1-4 register loads at capture event time.
7	CTRRST4		Counter Reset on Capture Event 4
		0	*Do not* reset counter on Capture Event 4 (absolute time stamp operation)
		1	Reset counter after Capture Event 4 time-stamp has been captured (used in difference mode operation)
6	CAP4POL		Capture Event 4 Polarity select
		0	Capture Event 4 triggered on a rising edge (RE)
		1	Capture Event 4 triggered on a falling edge (FE)
5	CTRRST3		Counter Reset on Capture Event 3
		0	*Do not* reset counter on Capture Event 3 (absolute time stamp)
		1	Reset counter after Event 3 time-stamp has been captured (used in difference mode operation)
4	CAP3POL		Capture Event 3 Polarity select
		0	Capture Event 3 triggered on a rising edge (RE)
		1	Capture Event 3 triggered on a falling edge (FE)
3	CTRRST2		Counter Reset on Capture Event 2
		0	*Do not* reset counter on Capture Event 2 (absolute time stamp)
		1	Reset counter after Event 2 time-stamp has been captured (used in difference mode operation)
2	CAP2POL		Capture Event 2 Polarity select
		0	Capture Event 2 triggered on a rising edge (RE)
		1	Capture Event 2 triggered on a falling edge (FE)
1	CTRRST1		Counter Reset on Capture Event 1
		0	*Do not* reset counter on Capture Event 1 (absolute time stamp)
		1	Reset counter after Event 1 time-stamp has been captured (used in difference mode operation)
0	CAP1POL		Capture Event 1 Polarity select
		0	Capture Event 1 triggered on a rising edge (RE)
		1	Capture Event 1 triggered on a falling edge (FE)

Figure 8-18. ECAP Control Register 2 (ECCTL2)

15	11	10	9	8
Reserved		APWMPOL	CAP/APWM	SWSYNC
R-0		R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
SYNCO_SEL		SYNCI_EN	TSCTRSTOP	REARM	STOP_WRAP		CONT/ONESHT
R/W-0		R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 8-8. ECAP Control Register 2 (ECCTL2) Field Descriptions

Bit(s)	Field		Description
15:11	Reserved		Reserved
10	APWMPOL		APWM output polarity select. This is applicable only in APWM operating mode
		0	Output is active high (Compare value defines high time)
		1	Output is active low (Compare value defines low time)
9	CAP/APWM		CAP/APWM operating mode select
		0	ECAP module operates in capture mode. This mode forces the following configuration: • Inhibits TSCTR resets via CTR = PRD event • Inhibits shadow loads on CAP1 and 2 registers • Permits user to enable CAP1-4 register load • CAPx/APWMx pin operates as a capture input
		1	ECAP module operates in APWM mode. This mode forces the following configuration: • Resets TSCTR on CTR = PRD event (period boundary • Permits shadow loading on CAP1 and 2 registers • Disables loading of time-stamps into CAP1-4 registers • CAPx/APWMx pin operates as a APWM output
8	SWSYNC		Software-forced Counter (TSCTR) Synchronizing. This provides a convenient software method to synchronize some or all ECAP time bases. In APWM mode, the synchronizing can also be done via the CTR = PRD event.
		0	Writing a zero has no effect. Reading always returns a zero
		1	Writing a one forces a TSCTR shadow load of current ECAP module and any ECAP modules down-stream providing the SYNCO_SEL bits are 0,0. After writing a 1, this bit returns to a zero.
			Note: Selection CTR = PRD is meaningful only in APWM mode; however, you can choose it in CAP mode if you find doing so useful.
7:6	SYNCO_SEL		Sync-Out Select
		00	Select sync-in event to be the sync-out signal (pass through)
		01	Select CTR = PRD event to be the sync-out signal
		10	Disable sync out signal
		11	Disable sync out signal
5	SYNCI_EN		Counter (TSCTR) Sync-In select mode
		0	Disable sync-in option
		1	Enable counter (TSCTR) to be loaded from CTRPHS register upon either a SYNCI signal or a S/W force event.
4	TSCTRSTOP		Time Stamp (TSCTR) Counter Stop (freeze) Control
		0	TSCTR stopped
		1	TSCTR free-running
3	RE-ARM		One-Shot Re-Arming Control (wait for stop trigger). Note: The re-arm function is valid in one shot or continuous mode.
		0	Has no effect (reading always returns a 0)
		1	Arms the one-shot sequence as follows: 1) Resets the Mod4 counter to zero 2) Unfreezes the Mod4 counter 3) Enables capture register loads
2:1	STOP_WRAP		Stop value for one-shot mode. This is the number (between 1-4) of captures allowed to occur before the CAP(1-4) registers are frozen, that is, capture sequence is stopped. Wrap value for continuous mode. This is the number (between 1-4) of the capture register in which the circular buffer wraps around and starts again.
		00	Stop after Capture Event 1 in one-shot mode Wrap after Capture Event 1 in continuous mode.
		01	Stop after Capture Event 2 in one-shot mode Wrap after Capture Event 2 in continuous mode.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 8-8. ECAP Control Register 2 (ECCTL2) Field Descriptions (continued)

Bit(s)	Field		Description
		10	Stop after Capture Event 3 in one-shot mode Wrap after Capture Event 3 in continuous mode.
		11	Stop after Capture Event 4 in one-shot mode Wrap after Capture Event 4 in continuous mode. Notes: STOP_WRAP is compared to Mod4 counter and, when equal, 2 actions occur: • Mod4 counter is stopped (frozen) • Capture register loads are inhibited In one-shot mode, further interrupt events are blocked until re-armed.
0	CONT/ONESHT		Continuous or one-shot mode control (applicable only in capture mode)
		0	Operate in continuous mode
		1	Operate in one-Shot mode

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 8-19. ECAP Interrupt Enable Register (ECEINT)

15							8
Reserved							
7	6	5	4	3	2	1	0
CTR=CMP	CTR=PRD	CTROVF	CEVT4	CEVT3	CEVT2	CETV1	Reserved
		R/W	R/W	R/W	R/W	R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-9. ECAP Interrupt Enable Register (ECEINT) Field Descriptions

Bits	Field	Value	Description
15:8	Reserved		
7	CTR=CMP		Counter Equal Compare Interrupt Enable
		0	Disable Compare Equal as an Interrupt source
		1	Enable Compare Equal as an Interrupt source
6	CTR=PRD		Counter Equal Period Interrupt Enable
		0	Disable Period Equal as an Interrupt source
		1	Enable Period Equal as an Interrupt source
5	CTROVF		Counter Overflow Interrupt Enable
		0	Disabled counter Overflow as an Interrupt source
		1	Enable counter Overflow as an Interrupt source
4	CEVT4		Capture Event 4 Interrupt Enable
		0	Disable Capture Event 4 as an Interrupt source
		1	Capture Event 4 Interrupt Enable
3	CEVT3		Capture Event 3 Interrupt Enable
		0	Disable Capture Event 3 as an Interrupt source
		1	Enable Capture Event 3 as an Interrupt source
2	CEVT2		Capture Event 2 Interrupt Enable
		0	Disable Capture Event 2 as an Interrupt source
		1	Enable Capture Event 2 as an Interrupt source
1	CEVT1		Capture Event 1 Interrupt Enable
		0	Disable Capture Event 1 as an Interrupt source
		1	Enable Capture Event 1 as an Interrupt source
0	Reserved		

The interrupt enable bits (CEVT1, ...) block any of the selected events from generating an interrupt. Events will still be latched into the flag bit (ECFLG register) and can be forced/cleared via the ECFRC/ECCLR registers.

The proper procedure for configuring peripheral modes and interrupts is as follows:

- Disable global interrupts
- Stop eCAP counter
- Disable eCAP interrupts
- Configure peripheral registers
- Clear spurious eCAP interrupt flags
- Enable eCAP interrupts
- Start eCAP counter
- Enable global interrupts

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 8-20. ECAP Interrupt Flag Register (ECFLG)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
CTR=CMP	CTR=PRD	CTROVF	CEVT4	CETV3	CEVT2	CETV1	INT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-10. ECAP Interrupt Flag Register (ECFLG) Field Descriptions

Bits	Field	Value	Description
15:8	Reserved		
7	CTR=CMP		Compare Equal Compare Status Flag. This flag is active only in APWM mode.
		0	Indicates no event occurred
		1	Indicates the counter (TSCTR) reached the compare register value (ACMP)
6	CTR=PRD		Counter Equal Period Status Flag. This flag is only active in APWM mode.
		0	Indicates no event occurred
		1	Indicates the counter (TSCTR) reached the period register value (APRD) and was reset.
5	CTROVF		Counter Overflow Status Flag. This flag is active in CAP and APWM mode.
		0	Indicates no event occurred.
		1	Indicates the counter (TSCTR) has made the transition from FFFFFFFF " 00000000
4	CEVT4		Capture Event 4 Status Flag This flag is only active in CAP mode.
		0	Indicates no event occurred
		1	Indicates the fourth event occurred at ECAPx pin
3	CEVT3		Capture Event 3 Status Flag. This flag is active only in CAP mode.
		0	Indicates no event occurred.
		1	Indicates the third event occurred at ECAPx pin.
2	CEVT2		Capture Event 2 Status Flag. This flag is only active in CAP mode.
		0	Indicates no event occurred.
		1	Indicates the second event occurred at ECAPx pin.
1	CEVT1		Capture Event 1 Status Flag. This flag is only active in CAP mode.
		0	Indicates no event occurred.
		1	Indicates the first event occurred at ECAPx pin.
0	INT		Global Interrupt Status Flag
		0	Indicates no interrupt generated.
		1	Indicates that an interrupt was generated.

Figure 8-21. ECAP Interrupt Clear Register (ECCLR)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
CTR=CMP	CTR=PRD	CTROVF	CEVT4	CETV3	CETV2	CETV1	INT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 8-11. ECAP Interrupt Clear Register (ECCLR) Field Descriptions

Bits	Field		Description
15:8	Reserved		Any writes to these bit(s) must always have a value of 0.
7	CTR=CMP		Counter Equal Compare Status Flag
		0	Writing a 0 has no effect. Always reads back a 0
		1	Writing a 1 clears the CTR=CMP flag condition
6	CTR=PRD		Counter Equal Period Status Flag
		0	Writing a 0 has no effect. Always reads back a 0
		1	Writing a 1 clears the CTR=PRD flag condition
5	CTROVF		Counter Overflow Status Flag
		0	Writing a 0 has no effect. Always reads back a 0
		1	Writing a 1 clears the CTROVF flag condition
4	CEVT4		Capture Event 4 Status Flag
		0	Writing a 0 has no effect. Always reads back a 0.
		1	Writing a 1 clears the CEVT4 flag condition.
3	CEVT3		Capture Event 3 Status Flag
		0	Writing a 0 has no effect. Always reads back a 0.
		1	Writing a 1 clears the CEVT3 flag condition.
2	CEVT2		Capture Event 2 Status Flag
		1	Writing a 0 has no effect. Always reads back a 0.
		0	Writing a 1 clears the CEVT2 flag condition.
1	CEVT1		Capture Event 1 Status Flag
		0	Writing a 0 has no effect. Always reads back a 0.
		1	Writing a 1 clears the CEVT1 flag condition.
0	INT		Global Interrupt Clear Flag
		0	Writing a 0 has no effect. Always reads back a 0.
		1	Writing a 1 clears the INT flag and enable further interrupts to be generated if any of the event flags are set to 1.

Figure 8-22. ECAP Interrupt Forcing Register (ECFRC)

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
CTR=CMP	CTR=PRD	CTROVF	CEVT4	CETV3	CETV2	CETV1	reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8-12. ECAP Interrupt Forcing Register (ECFRC) Field Descriptions

Bits	Field	Value	Description
15:8	Reserved	0	Any writes to these bit(s) must always have a value of 0.
7	CTR=CMP		Force Counter Equal Compare Interrupt
		0	No effect. Always reads back a 0.
		1	Writing a 1 sets the CTR=CMP flag bit.
6	CTR=PRD		Force Counter Equal Period Interrupt
		0	No effect. Always reads back a 0.
		1	Writing a 1 sets the CTR=PRD flag bit.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 8-12. ECAP Interrupt Forcing Register (ECFRC) Field Descriptions (continued)

Bits	Field	Value	Description
5	CTROVF		Force Counter Overflow
		0	No effect. Always reads back a 0.
		1	Writing a 1 to this bit sets the CTROVF flag bit.
4	CEVT4		Force Capture Event 4
		0	No effect. Always reads back a 0.
		1	Writing a 1 sets the CEVT4 flag bit
3	CEVT3		Force Capture Event 3
		0	No effect. Always reads back a 0.
		1	Writing a 1 sets the CEVT3 flag bit
2	CEVT2		Force Capture Event 2
		0	No effect. Always reads back a 0.
		1	Writing a 1 sets the CEVT2 flag bit.
1	CEVT1		Force Capture Event 1
		1	No effect. Always reads back a 0.
		0	Sets the CEVT1 flag bit.
0	Reserved	0	Any writes to these bit(s) must always have a value of 0.

8.6 Register Mapping

Table 8-13 shows the eCAP module control and status register set.

Table 8-13. Control and Status Register Set

Name	Offset	Size (x16)	Description
Time Base Module Registers			
TSCTR	0x0000	2	Time-Stamp Counter
CTRPHS	0x0002	2	Counter Phase Offset Value Register
CAP1	0x0004	2	Capture 1 Register
CAP2	0x0006	2	Capture 2 Register
CAP3	0x0008	2	Capture 3 Register
CAP4	0x000A	2	Capture 4 Register
reserved	0x000C - 0x0013	8	
ECCTL1	0x0014	1	Capture Control Register 1
ECCTL2	0x0015	1	Capture Control Register 2
ECEINT	0x0016	1	Capture Interrupt Enable Register
ECFLG	0x0017	1	Capture Interrupt Flag Register
ECCLR	0x0018	1	Capture Interrupt Clear Register
ECFRC	0x0019	1	Capture Interrupt Force Register
Reserved	0x001A - 0x001F	6	

8.7 Application of the ECAP Module

The following sections will provide Applications examples and code snippets to show how to configure and operate the eCAP module. For clarity and ease of use, the examples use the eCAP “C” header files. Below are useful #defines which will help in the understanding of the examples.

```
// ECCTL1 (ECAP Control Reg 1)

 //=========================
```


Copyright © 2012–2019, Texas Instruments Incorporated

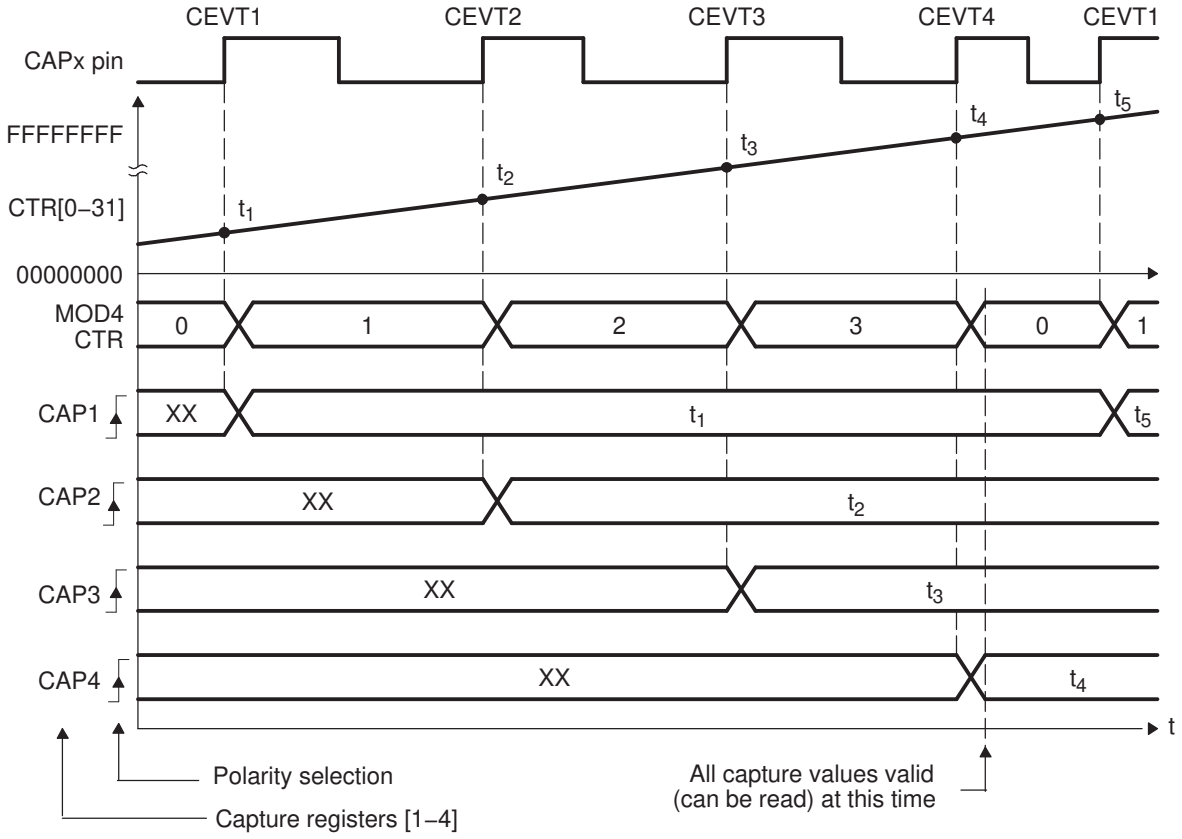
```
// CAPxPOL bits      #define EC_RISING 0x0    #define EC_FALLING 0x1
// CTRRSTx bits      #define EC_ABS_MODE 0x0  #define EC_DELTA_MODE 0x1
// PRESCALE bits     #define EC_BYPASS 0x0    #define EC_DIV1 0x0 #define EC_DIV2 0x1
                                              #define EC_DIV4 0x2 #define EC_DIV6 0x3
                                              #define EC_DIV8 0x4 #define EC_DIV10 0x5
// ECCTL2 ( ECAP Control Reg 2)
//==========================
// CONT/ONESHOT bit  #define EC_CONTINUOUS 0x0 #define EC_ONESHOT 0x1
// STOPVALUE bit     #define EC_EVENT1 0x0     #define EC_EVENT2 0x1 #define EC_EVENT3 0x2
                                               #define EC_EVENT4 0x3
// RE-ARM bit #define EC_ARM 0x1
// TSCTRSTOP bit     #define EC_FREEZE 0x0     #define EC_RUN 0x1
// SYNCO_SEL bit     #define EC_SYNCIN 0x0     #define EC_CTR_PRD 0x1
                                               #define EC_SYNCO_DIS 0x2
// CAP/APWM mode bit #define EC_CAP_MODE 0x0   #define EC_APWM_MODE 0x1
// APWMPOL bit       #define EC_ACTV_HI 0x0    #define EC_ACTV_LO 0x1
// Generic           #define EC_DISABLE 0x0    #define EC_ENABLE 0x1 #define EC_FORCE 0x1
```

8.7.1 Example 1 - Absolute Time-Stamp Operation Rising Edge Trigger

Figure 8-23 shows an example of continuous capture operation (Mod4 counter wraps around). In this figure, TSCTR counts-up without resetting and capture events are qualified on the rising edge only, this gives period (and frequency) information.

On an event, the TSCTR contents (time-stamp) is first captured, then Mod4 counter is incremented to the next state. When the TSCTR reaches FFFFFFFF (maximum value), it wraps around to 00000000 (not shown in Figure 8-23), if this occurs, the CTROVF (counter overflow) flag is set, and an interrupt (if enabled) occurs, CTROVF (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. Captured Time-stamps are valid at the point indicated by the diagram (after the 4th event), hence event CEVT4 can conveniently be used to trigger an interrupt and the CPU can read data from the CAPx registers.

Figure 8-23. Capture Sequence for Absolute Time-stamp and Rising Edge Detect



Copyright © 2012–2019, Texas Instruments Incorporated

8.7.1.1 Code snippet for CAP mode Absolute Time, Rising Edge Trigger

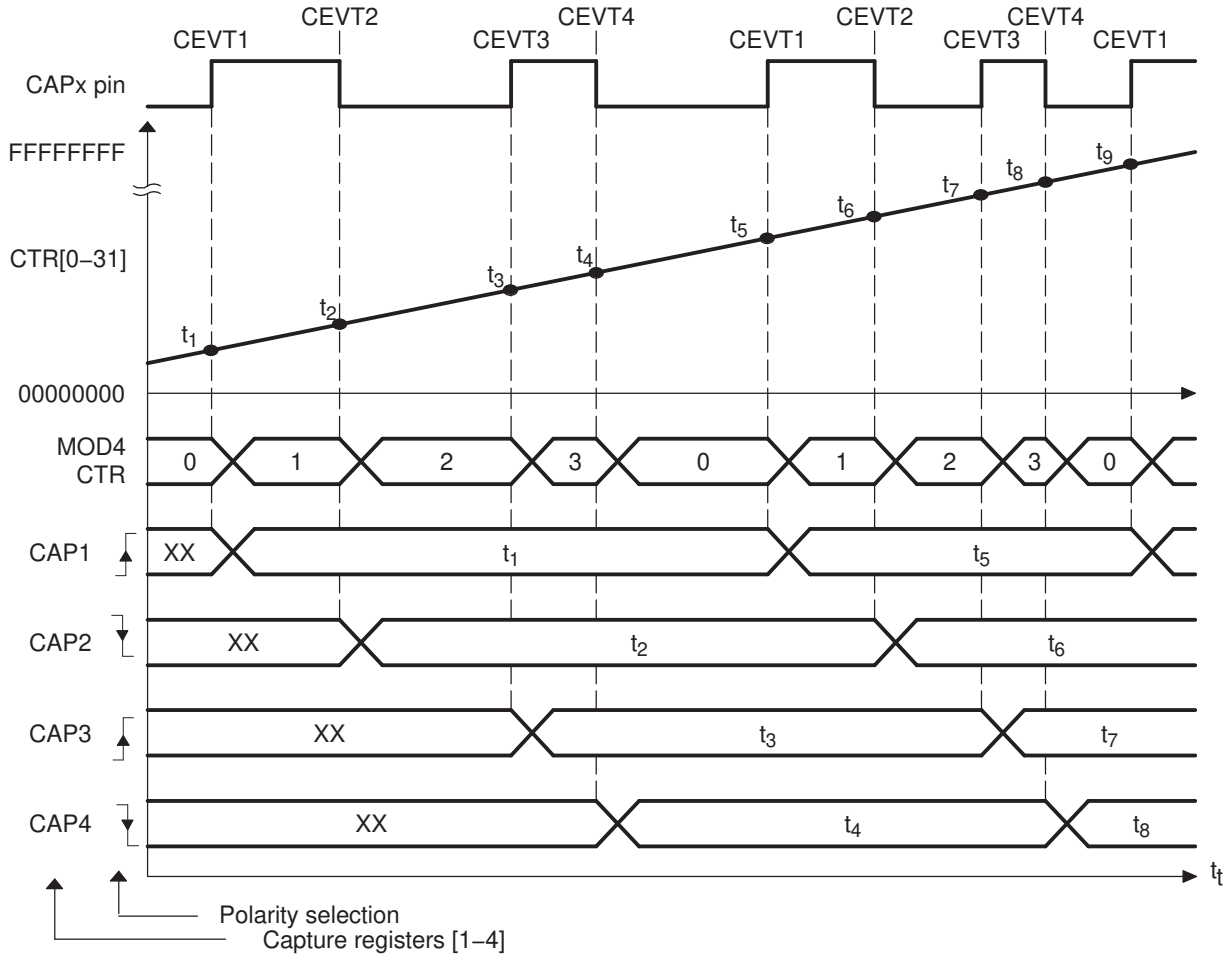
```
// Code snippet for CAP mode Absolute Time, Rising edge trigger

// Initialization Time
//=======================

// ECAP module 1 config ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;

   ECap1Regs.ECCTL1.bit.CAP2POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP4POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CTRRST1 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST2 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST3 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST4 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
   ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
   ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
   ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
   ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
   ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
   ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT4 triggered ISR call)
//=========================================

TSt1 = ECap1Regs.CAP1;
// Fetch Time-Stamp captured at t1 TSt2 = ECap1Regs.CAP2;
// Fetch Time-Stamp captured at t2 TSt3 = ECap1Regs.CAP3;
// Fetch Time-Stamp captured at t3 TSt4 = ECap1Regs.CAP4;
// Fetch Time-Stamp captured at t4 Period1 = TSt2-TSt1;
// Calculate 1st period Period2 = TSt3-TSt2;
// Calculate 2nd period Period3 = TSt4-TSt3;
// Calculate 3rd period
```


Copyright © 2012–2019, Texas Instruments Incorporated

8.7.2 Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger

In Figure 8-24 the eCAP operating mode is almost the same as in the previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information, i.e: Period1 = $t_3 - t_1$, Period2 = $t_5 - t_3$, …etc. Duty Cycle1 (on-time %) = $(t_2 - t_1)$ / Period1 x 100%, etc. Duty Cycle1 (off-time %) = $(t_3 - t_2)$ / Period1 x 100%, etc.

Figure 8-24. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect



Copyright © 2012–2019, Texas Instruments Incorporated

8.7.2.1 Code Snippet for CAP mode Absolute Time, Rising and Falling Edge Triggers

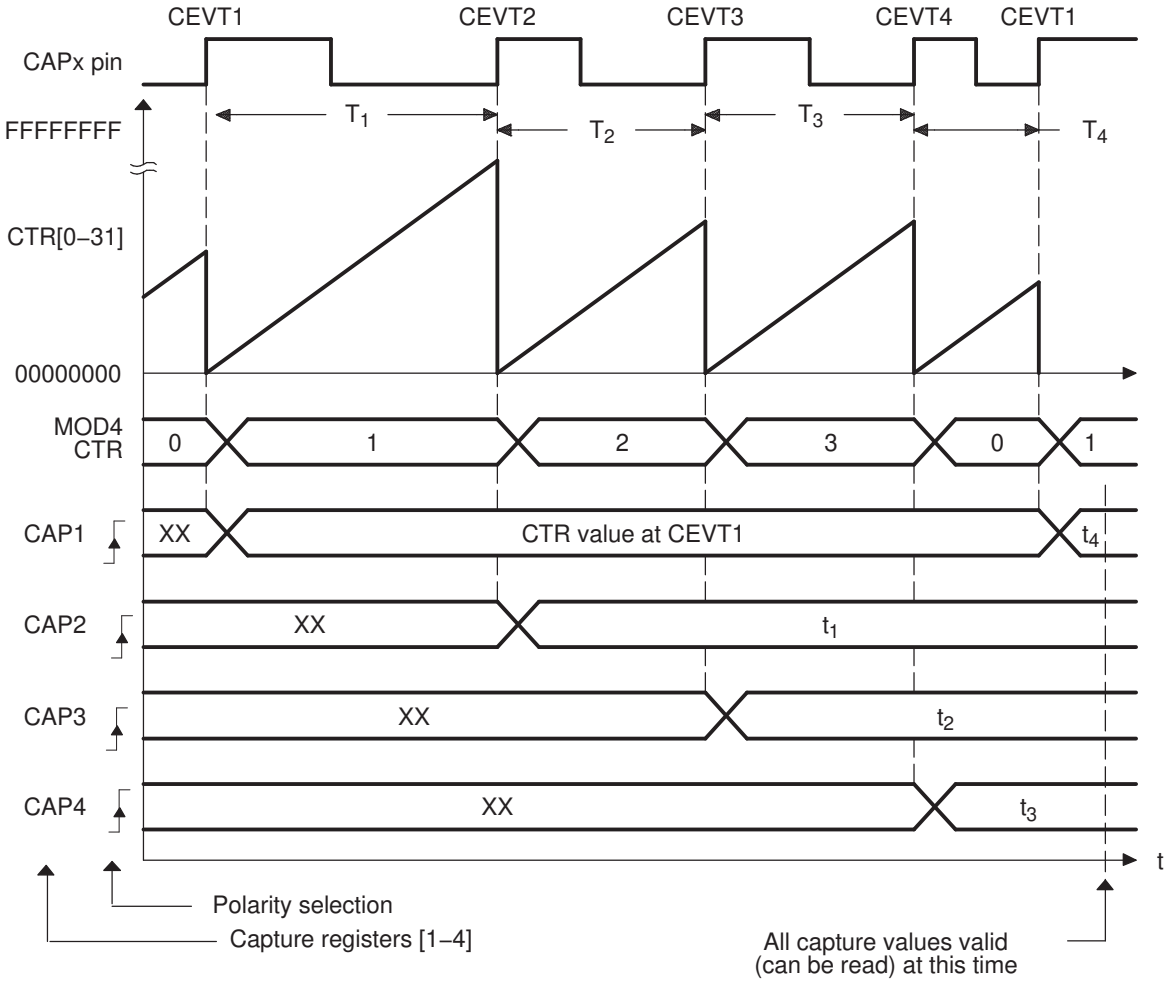
```
// Code snippet for CAP mode Absolute Time, Rising and Falling

// edge triggers // Initialization Time

//=======================

// ECAP module 1 config ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP2POL = EC_FALLING;
   ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP4POL = EC_FALLING;
   ECap1Regs.ECCTL1.bit.CTRRST1 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST2 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST3 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST4 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
   ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
   ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
   ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
   ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
   ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
   ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT4 triggered ISR call)

//=========================================

TSt1 = ECap1Regs.CAP1;
// Fetch Time-Stamp captured at t1 TSt2 = ECap1Regs.CAP2;
// Fetch Time-Stamp captured at t2 TSt3 = ECap1Regs.CAP3;
// Fetch Time-Stamp captured at t3 TSt4 = ECap1Regs.CAP4;
// Fetch Time-Stamp captured at t4 Period1 = TSt3-TSt1;
// Calculate 1st period DutyOnTime1 = TSt2-TSt1;
// Calculate On time DutyOffTime1 = TSt3-TSt2;
// Calculate Off time
```


Copyright © 2012–2019, Texas Instruments Incorporated

8.7.3 Example 3 - Time Difference (Delta) Operation Rising Edge Trigger

This example Figure 8-25 shows how the eCAP module can be used to collect Delta timing data from pulse train waveforms. Here Continuous Capture mode (TSCTR counts-up without resetting, and Mod4 counter wraps around) is used. In Delta-time mode, TSCTR is Reset back to Zero on every valid event. Here Capture events are qualified as Rising edge only. On an event, TSCTR contents (Time-Stamp) is captured first, and then TSCTR is reset to Zero. The Mod4 counter then increments to the next state. If TSCTR reaches FFFFFFFF (Max value), before the next event, it wraps around to 00000000 and continues, a CNTOVF (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. The advantage of Delta-time Mode is that the CAPx contents directly give timing data without the need for CPU calculations, that is, Period1 = T_1, Period2 = T_2,…etc. As shown in the diagram, the CEVT1 event is a good trigger point to read the timing data, T_1, T_2, T_3, T_4 are all valid here.

Figure 8-25. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect



Copyright © 2012–2019, Texas Instruments Incorporated

8.7.3.1 Code snippet for CAP mode Delta Time, Rising Edge Trigger

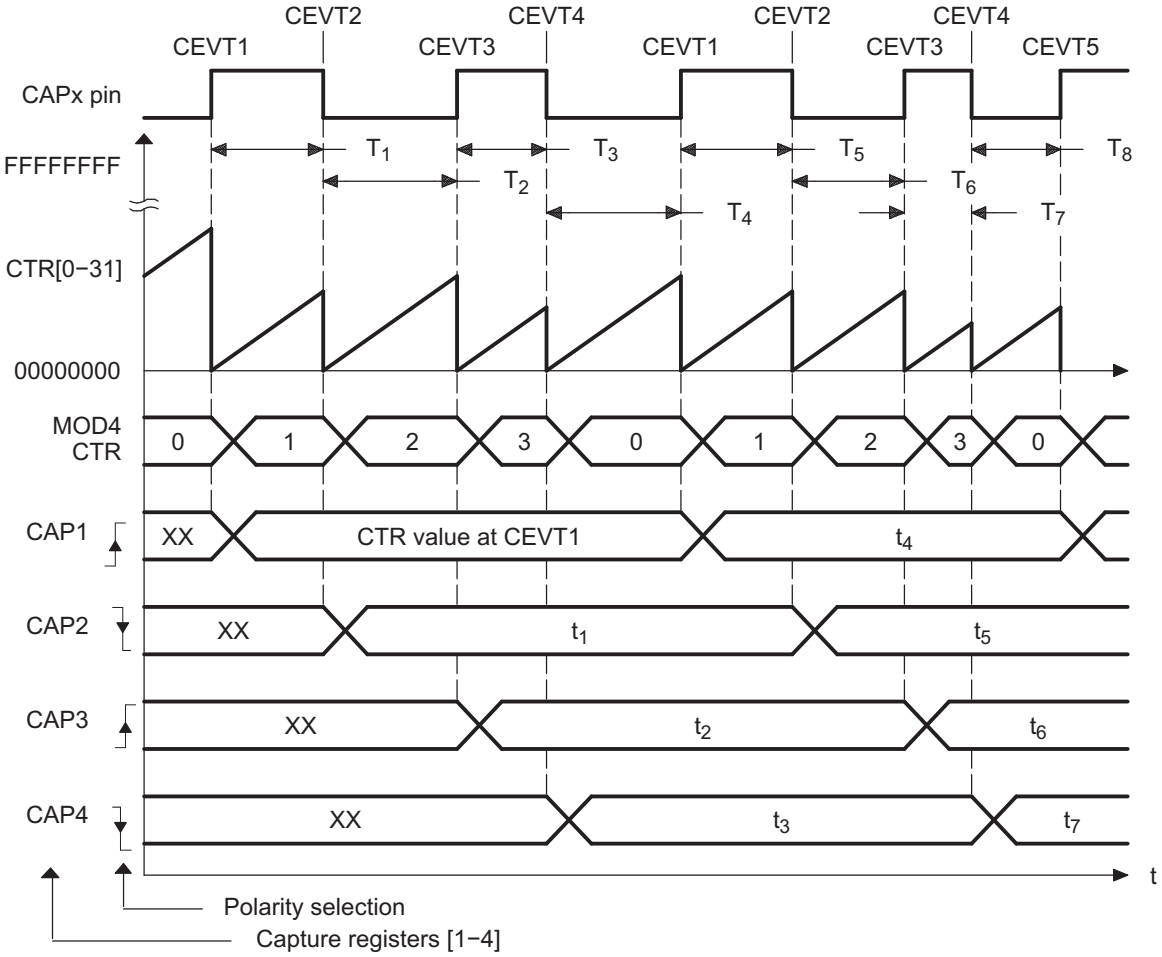
```
// Code snippet for CAP mode Delta Time, Rising edge trigger

// Initialization Time

//=======================

// ECAP module 1 config ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP2POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP4POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CTRRST1 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST2 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST3 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST4 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
   ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
   ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
   ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
   ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
   ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
   ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT1 triggered ISR call)

//=========================================

// Note: here Time-stamp directly represents the Period value. Period4 = ECap1Regs.CAP1;
// Fetch Time-Stamp captured at T1 Period1 = ECap1Regs.CAP2;
// Fetch Time-Stamp captured at T2 Period2 = ECap1Regs.CAP3;
// Fetch Time-Stamp captured at T3 Period3 = ECap1Regs.CAP4;
// Fetch Time-Stamp captured at T4
```


Copyright © 2012–2019, Texas Instruments Incorporated

8.7.4 Example 4 - Time Difference (Delta) Operation Rising and Falling Edge Trigger

In Figure 8-26 the eCAP operating mode is almost the same as in previous section except Capture events are qualified as either Rising or Falling edge, this now gives both Period and Duty cycle information, i.e: Period1 = T_1+T_2, Period2 = T_3+T_4, …etc Duty Cycle1 (on-time %) = T_1 / Period1 x 100%, etc Duty Cycle1 (off-time %) = T_2 / Period1 x 100%, etc

Figure 8-26. Capture Sequence for Delta Mode Time-stamp With Rising and Falling Edge Detect



During initialization, you must write to the active registers for both period and compare. This will then automatically copy the init values into the shadow values. For subsequent compare updates, during run-time, only the shadow registers must be used.

Copyright © 2012–2019, Texas Instruments Incorporated

8.7.4.1 Code snippet for CAP mode Delta Time, Rising and Falling Edge Triggers

```
// Code snippet for CAP mode Delta Time, Rising and Falling
// edge triggers
// Initialization Time

//=======================

// ECAP module 1 config ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP2POL = EC_FALLING;
   ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP4POL = EC_FALLING;
   ECap1Regs.ECCTL1.bit.CTRRST1 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST2 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST3 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST4 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
   ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
   ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
   ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
   ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
   ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
   ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT1 triggered ISR call)

//=========================================
//
Note: here Time-stamp directly represents the Duty cycle values. DutyOnTime1 = ECap1Regs.CAP2;

// Fetch Time-Stamp captured at T2 DutyOffTime1 = ECap1Regs.CAP3;
// Fetch Time-Stamp captured at T3 DutyOnTime2 = ECap1Regs.CAP4;
// Fetch Time-Stamp captured at T4 DutyOffTime2 = ECap1Regs.CAP1;
// Fetch Time-
Stamp captured at T1 Period1 = DutyOnTime1 + DutyOffTime1; Period2 = DutyOnTime2 + DutyOffTime2;
```

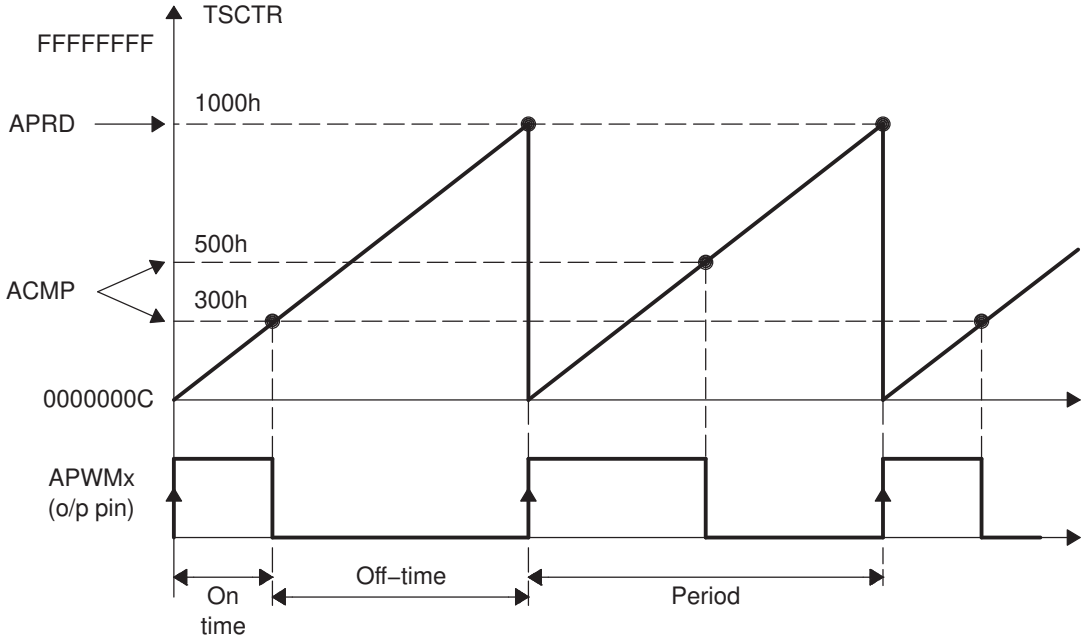

Copyright © 2012–2019, Texas Instruments Incorporated

8.8 Application of the APWM Mode

In this example, the eCAP module is configured to operate as a PWM generator. Here a very simple single channel PWM waveform is generated from output pin APWMx. The PWM polarity is active high, which means that the compare value (CAP2 reg is now a compare register) represents the on-time (high level) of the period. Alternatively, if the APWMPOL bit is configured for active low, then the compare value represents the off-time. Note here values are in hexadecimal (“h”) notation.

8.8.1 Example 1 - Simple PWM Generation (Independent Channel/s)

Figure 8-27. PWM Waveform Details of APWM Mode Operation



Example 8-1. Code Snippet for APWM Mode

```
// Code snippet for APWM mode Example 1

// Initialization Time

//=======================

// ECAP module 1 config ECap1Regs.CAP1 = 0x1000;

// Set period value ECap1Regs.CTRPHS = 0x0;

// make phase zero ECap1Regs.ECCTL2.bit.CAP_APWM = EC_APWM_MODE;
                   ECap1Regs.ECCTL2.bit.APWMPOL = EC_ACTV_HI;

// Active high ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;

// Synch not used ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;

// Synch not used ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (Instant 1, for example, ISR call)

//======================
   ECap1Regs.CAP2 = 0x300;
```


Copyright © 2012–2019, Texas Instruments Incorporated

Example 8-1. Code Snippet for APWM Mode (continued)

```
// Set Duty cycle, that  is,  compare value

// Run Time (Instant 2, for example, another ISR call)

//======================\
   ECap1Regs.CAP2 = 0x500;

// Set Duty cycle, that is, compare value
```


Copyright © 2012–2019, Texas Instruments Incorporated

C28 Enhanced QEP (eQEP) Module

The eQEP module described here is a Type 0 eQEP. See the *TMS320x28xx, 28xxx DSP Peripheral Reference Guide* (SPRU566) for a list of all devices with a module of the same type to determine the differences between types and for a list of device-specific differences within a type.

The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in a high-performance motion and position-control system.

Topic **Page**

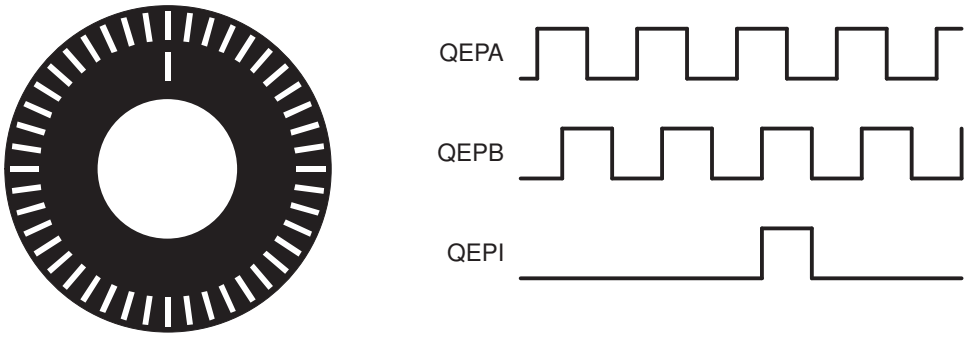
9.1 Introduction **821**
9.2 Description **823**
9.3 Quadrature Decoder Unit (QDU) **826**
9.4 Position Counter and Control Unit (PCCU) **829**
9.5 eQEP Edge Capture Unit **835**
9.6 eQEP Watchdog **839**
9.7 Unit Timer Base **839**
9.8 eQEP Interrupt Structure **840**
9.9 eQEP Registers **840**

Copyright © 2012–2019, Texas Instruments Incorporated

9.1 Introduction

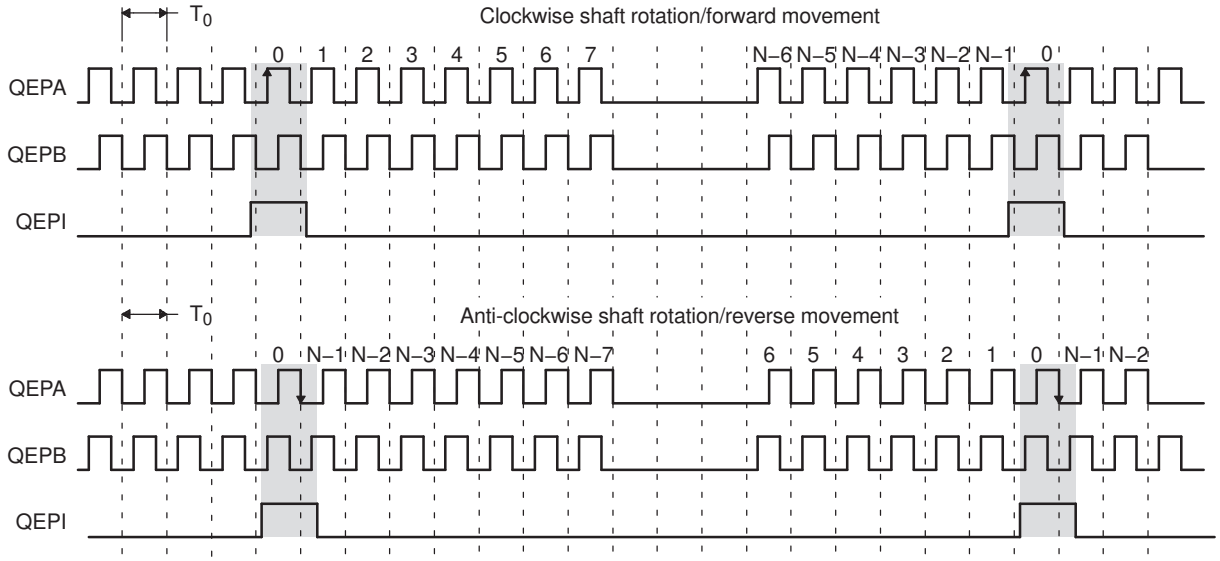
A single track of slots patterns the periphery of an incremental encoder disk, as shown in Figure 9-1. These slots create an alternating pattern of dark and light lines. The disk count is defined as the number of dark/light line pairs that occur per revolution (lines per revolution). As a rule, a second track is added to generate a signal that occurs once per revolution (index signal: QEPI), which can be used to indicate an absolute position. Encoder manufacturers identify the index pulse using different terms such as index, marker, home position, and zero reference

Figure 9-1. Optical Encoder Disk



To derive direction information, the lines on the disk are read out by two different photo-elements that "look" at the disk pattern with a mechanical shift of 1/4 the pitch of a line pair between them. This shift is realized with a reticle or mask that restricts the view of the photo-element to the desired part of the disk lines. As the disk rotates, the two photo-elements generate signals that are shifted 90° out of phase from each other. These are commonly called the quadrature QEPA and QEPB signals. The clockwise direction for most encoders is defined as the QEPA channel going positive before the QEPB channel and vise versa as shown in Figure 9-2.

Figure 9-2. QEP Encoder Output Signal for Forward/Reverse Movement



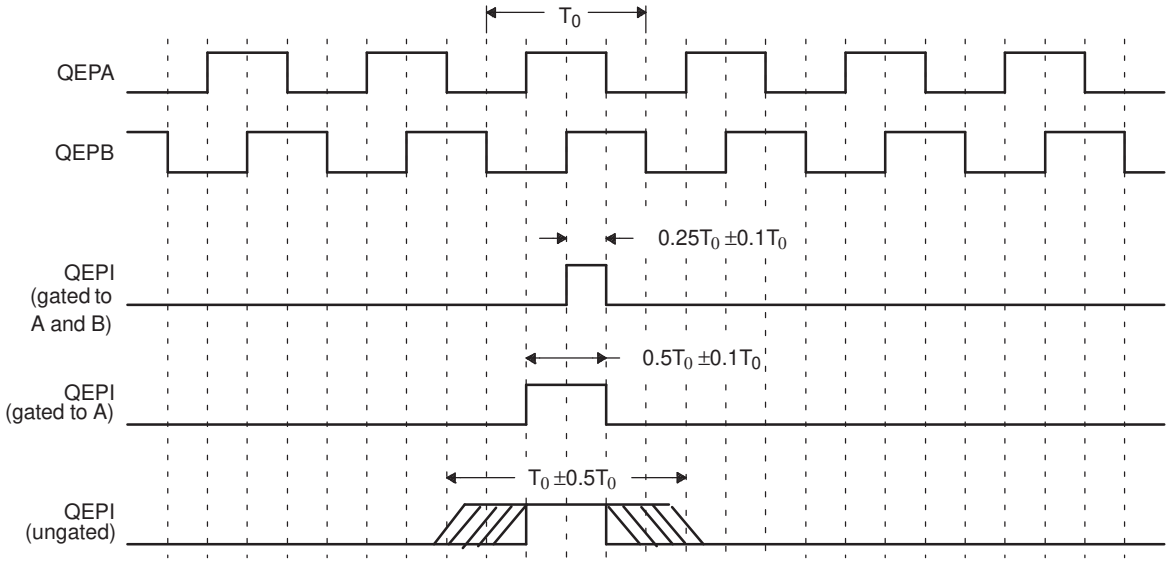
Legend: N = lines per revolution

The encoder wheel typically makes one revolution for every revolution of the motor or the wheel may be at a geared rotation ratio with respect to the motor. Therefore, the frequency of the digital signal coming from the QEPA and QEPB outputs varies proportionally with the velocity of the motor. For example, a 2000-line encoder directly coupled to a motor running at 5000 revolutions per minute (rpm) results in a frequency of 166.6 KHz, so by measuring the frequency of either the QEPA or QEPB output, the processor can determine the velocity of the motor.

Copyright © 2012–2019, Texas Instruments Incorporated

Quadrature encoders from different manufacturers come with two forms of index pulse (gated index pulse or ungated index pulse) as shown in Figure 9-3. A nonstandard form of index pulse is ungated. In the ungated configuration, the index edges are not necessarily coincident with A and B signals. The gated index pulse is aligned to any of the four quadrature edges and width of the index pulse and can be equal to a quarter, half, or full period of the quadrature signal.

Figure 9-3. Index Pulse Example



Some typical applications of shaft encoders include robotics and even computer input in the form of a mouse. Inside your mouse you can see where the mouse ball spins a pair of axles (a left/right, and an up/down axle). These axles are connected to optical shaft encoders that effectively tell the computer how fast and in what direction the mouse is moving.

General Issues: Estimating velocity from a digital position sensor is a cost-effective strategy in motor control. Two different first order approximations for velocity may be written as:

$$v(k) \approx \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (1)$$

$$v(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (2)$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k-1

T: Fixed unit time or inverse of velocity calculation rate

ΔX: Incremental position movement in unit time

t(k): Time instant "k"

t(k-1): Time instant "k-1"

X: Fixed unit position

ΔT: Incremental time elapsed for unit position movement.

Equation 1 is the conventional approach to velocity estimation and it requires a time base to provide unit time event for velocity calculation. Unit time is basically the inverse of the velocity calculation rate.

Copyright © 2012–2019, Texas Instruments Incorporated

The encoder count (position) is read once during each unit time event. The quantity [x(k) - x(k-1)] is formed by subtracting the previous reading from the current reading. Then the velocity estimate is computed by multiplying by the known constant 1/T (where T is the constant time between unit time events and is known in advance).

Estimation based on Equation 1 has an inherent accuracy limit directly related to the resolution of the position sensor and the unit time period T. For example, consider a 500-line per revolution quadrature encoder with a velocity calculation rate of 400 Hz. When used for position the quadrature encoder gives a four-fold increase in resolution, in this case, 2000 counts per revolution. The minimum rotation that can be detected is therefore 0.0005 revolutions, which gives a velocity resolution of 12 rpm when sampled at 400 Hz. While this resolution may be satisfactory at moderate or high speeds, e.g. 1% error at 1200 rpm, it would clearly prove inadequate at low speeds. In fact, at speeds below 12 rpm, the speed estimate would erroneously be zero much of the time.

At low speed, Equation 2 provides a more accurate approach. It requires a position sensor that outputs a fixed interval pulse train, such as the aforementioned quadrature encoder. The width of each pulse is defined by motor speed for a given sensor resolution. Equation 2 can be used to calculate motor speed by measuring the elapsed time between successive quadrature pulse edges. However, this method suffers from the opposite limitation, as does Equation 1. A combination of relatively large motor speeds and high sensor resolution makes the time interval ΔT small, and thus more greatly influenced by the timer resolution. This can introduce considerable error into high-speed estimates.

For systems with a large speed range (that is, speed estimation is needed at both low and high speeds), one approach is to use Equation 2 at low speed and have the DSP software switch over to Equation 1 when the motor speed rises above some specified threshold.

9.2 Description

This section provides the eQEP inputs, memory map, and functional description.

9.2.1 EQEP Inputs

The eQEP inputs include two pins for quadrature-clock mode or direction-count mode, an index (or 0 marker), and a strobe input. The eQEP module requires that the QEPA, QEPB, and QEPI inputs are synchronized to SYSCLK prior to entering the module. The application code should enable the synchronous GPIO input feature on any eQEP-enabled GPIO pins (see the *System Control and Interrupts*chapter for more details).

- *QEPA/XCLK and QEPB/XDIR*

  These two pins can be used in quadrature-clock mode or direction-count mode.

  – *Quadrature-clock Mode*

    The eQEP encoders provide two square wave signals (A and B) 90 electrical degrees out of phase whose phase relationship is used to determine the direction of rotation of the input shaft and number of eQEP pulses from the index position to derive the relative position information. For forward or clockwise rotation, QEPA signal leads QEPB signal and vice versa. The quadrature decoder uses these two inputs to generate quadrature-clock and direction signals.

  – *Direction-count Mode*

    In direction-count mode, direction and clock signals are provided directly from the external source. Some position encoders have this type of output instead of quadrature output. The QEPA pin provides the clock input and the QEPB pin provides the direction input.

- *QEPI: Index or Zero Marker*

  The eQEP encoder uses an index signal to assign an absolute start position from which position information is incrementally encoded using quadrature pulses. This pin is connected to the index output of the eQEP encoder to optionally reset the position counter for each revolution. This signal can be used to initialize or latch the position counter on the occurrence of a desired event on the index pin.

- QEPS: *Strobe Input*

  This general-purpose strobe signal can initialize or latch the position counter on the occurrence of a desired event on the strobe pin. This signal is typically connected to a sensor or limit switch to notify that the motor has reached a defined position.

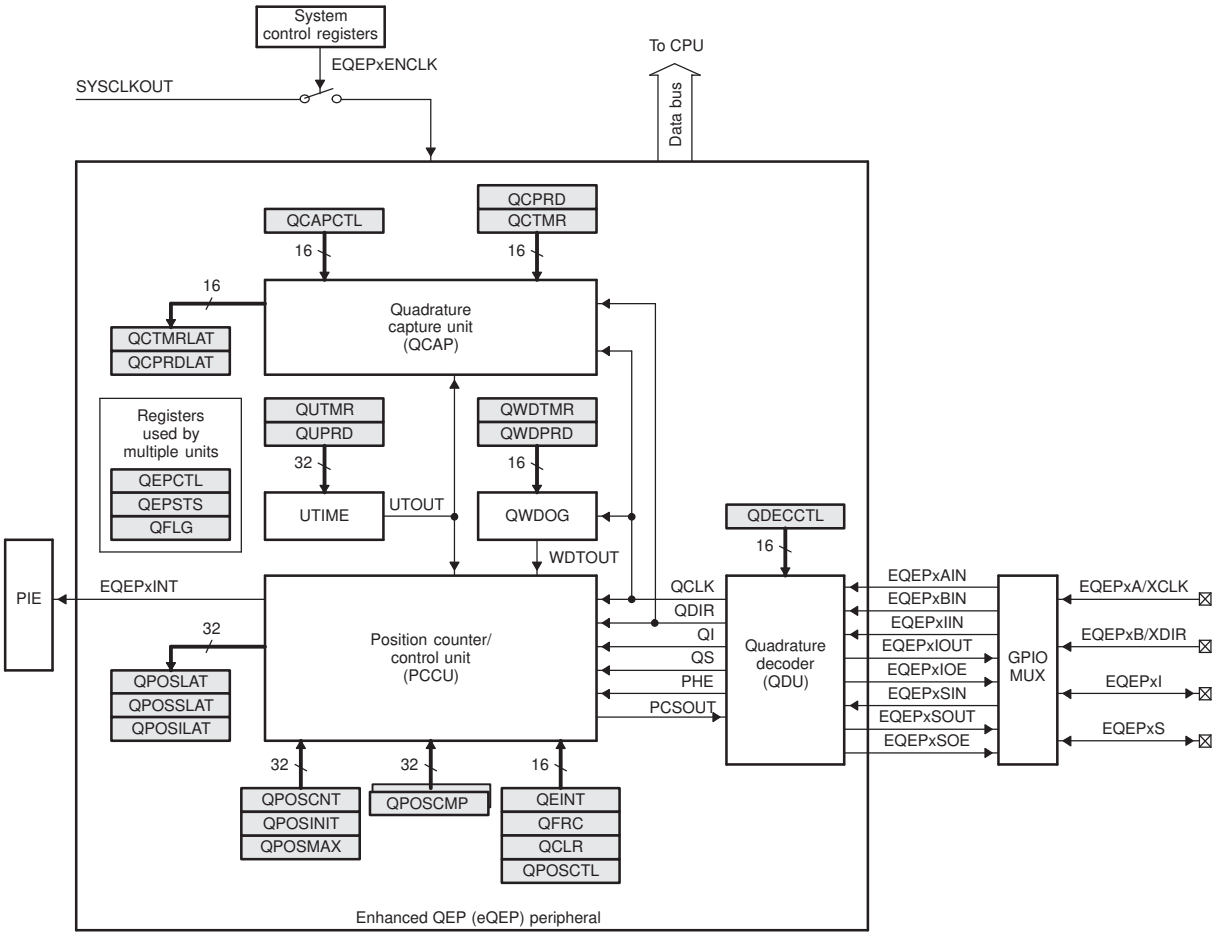
Copyright © 2012–2019, Texas Instruments Incorporated

9.2.2 Functional Description

The eQEP peripheral contains the following major functional units (as shown in Figure 9-4):

- Programmable input qualification for each pin (part of the GPIO MUX)
- Quadrature decoder unit (QDU)
- Position counter and control unit for position measurement (PCCU)
- Quadrature edge-capture unit for low-speed measurement (QCAP)
- Unit time base for speed/frequency measurement (UTIME)
- Watchdog timer for detecting stalls (QWDOG)

Figure 9-4. Functional Block Diagram of the eQEP Peripheral



9.2.3 eQEP Memory Map

Table 9-1 lists the registers with their memory locations, sizes, and reset values.

Table 9-1. EQEP Memory Map

Name	Offset	Size(x16)/ #shadow	Reset	Register Description
QPOSCNT	0x00	2/0	0x00000000	eQEP Position Counter
QPOSINIT	0x02	2/0	0x00000000	eQEP Initialization Position Count
QPOSMAX	0x04	2/0	0x00000000	eQEP Maximum Position Count
QPOSCMP	0x06	2/1	0x00000000	eQEP Position-compare
QPOSILAT	0x08	2/0	0x00000000	eQEP Index Position Latch

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-1. EQEP Memory Map (continued)

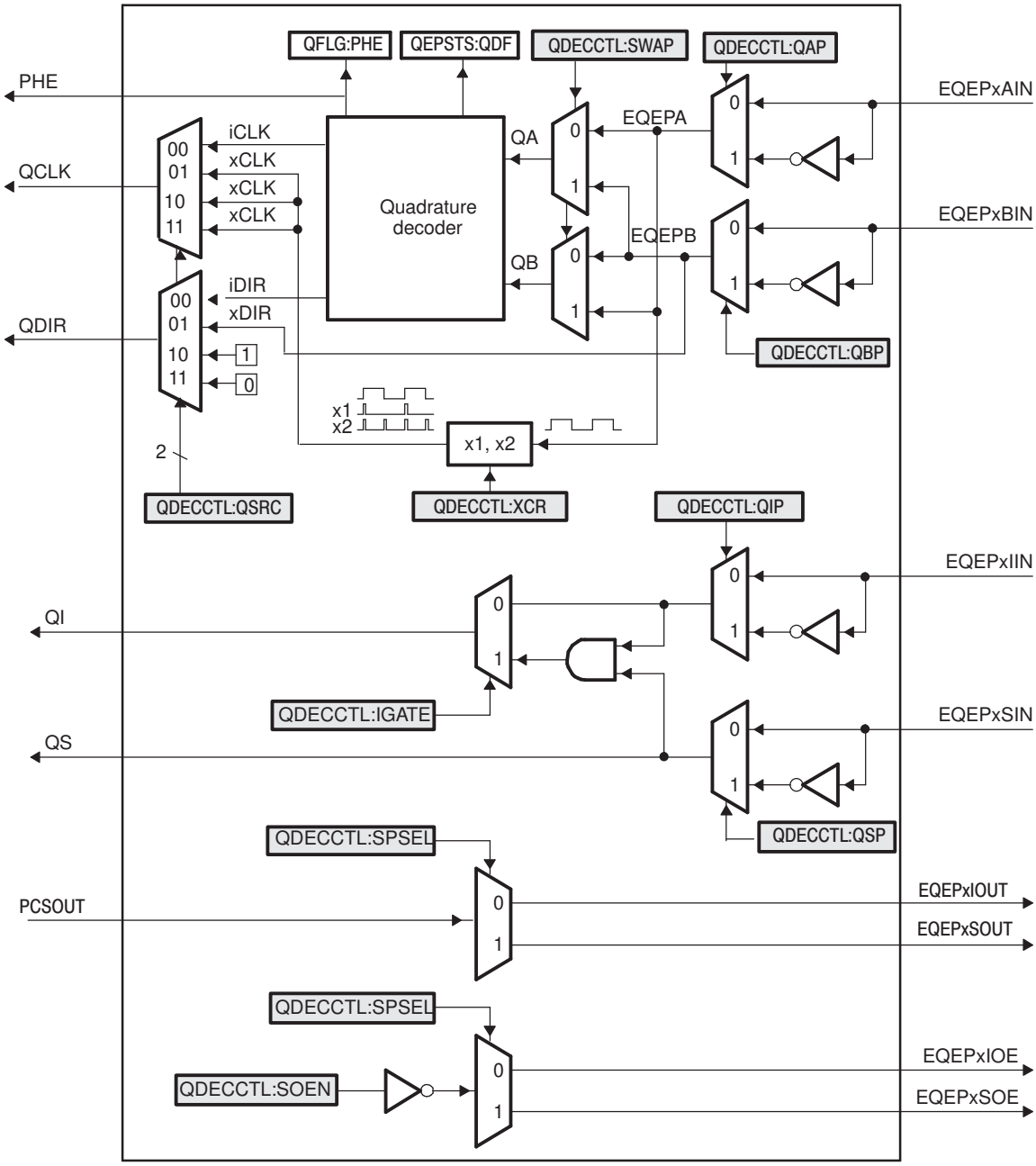
Name	Offset	Size(x16)/ #shadow	Reset	Register Description
QPOSSLAT	0x0A	2/0	0x00000000	eQEP Strobe Position Latch
QPOSLAT	0x0C	2/0	0x00000000	eQEP Position Latch
QUTMR	0x0E	2/0	0x00000000	QEP Unit Timer
QUPRD	0x10	2/0	0x00000000	eQEP Unit Period Register
QWDTMR	0x12	1/0	0x0000	eQEP Watchdog Timer
QWDPRD	0x13	1/0	0x0000	eQEP Watchdog Period Register
QDECCTL	0x14	1/0	0x0000	eQEP Decoder Control Register
QEPCTL	0x15	1/0	0x0000	eQEP Control Register
QCAPCTL	0x16	1/0	0x0000	eQEP Capture Control Register
QPOSCTL	0x17	1/0	0x00000	eQEP Position-compare Control Register
QEINT	0x18	1/0	0x0000	eQEP Interrupt Enable Register
QFLG	0x19	1/0	0x0000	eQEP Interrupt Flag Register
QCLR	0x1A	1/0	0x0000	eQEP Interrupt Clear Register
QFRC	0x1B	1/0	0x0000	eQEP Interrupt Force Register
QEPSTS	0x1C	1/0	0x0000	eQEP Status Register
QCTMR	0x1D	1/0	0x0000	eQEP Capture Timer
QCPRD	0x1E	1/0	0x0000	eQEP Capture Period Register
QCTMRLAT	0x1F	1/0	0x0000	eQEP Capture Timer Latch
QCPRDLAT	0x20	1/0	0x0000	eQEP Capture Period Latch
reserved	0x21 to 0x3F	31/0		

Copyright © 2012–2019, Texas Instruments Incorporated

9.3 Quadrature Decoder Unit (QDU)

Figure 9-5 shows a functional block diagram of the QDU.

Figure 9-5. Functional Block Diagram of Decoder Unit



9.3.1 Position Counter Input Modes

Clock and direction input to position counter is selected using QDECCTL[QSRC] bits, based on interface input requirement as follows:

- Quadrature-count mode
- Direction-count mode
- UP-count mode
- DOWN-count mode

Copyright © 2012–2019, Texas Instruments Incorporated

9.3.1.1 Quadrature Count Mode

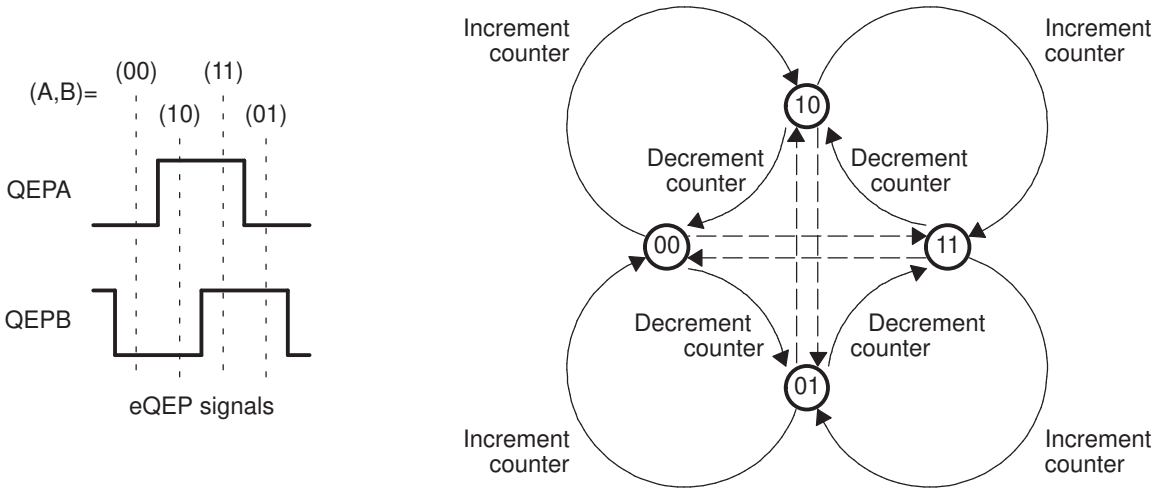
The quadrature decoder generates the direction and clock to the position counter in quadrature count mode.

Direction Decoding— The direction decoding logic of the eQEP circuit determines which one of the sequences (QEPA, QEPB) is the leading sequence and accordingly updates the direction information in QEPSTS[QDF] bit. Table 9-2 and Figure 9-6 show the direction decoding logic in truth table and state machine form. Both edges of the QEPA and QEPB signals are sensed to generate count pulses for the position counter. Therefore, the frequency of the clock generated by the eQEP logic is four times that of each input sequence. Figure 9-7 shows the direction decoding and clock generation from the eQEP input signals.

Table 9-2. Quadrature Decoder Truth Table

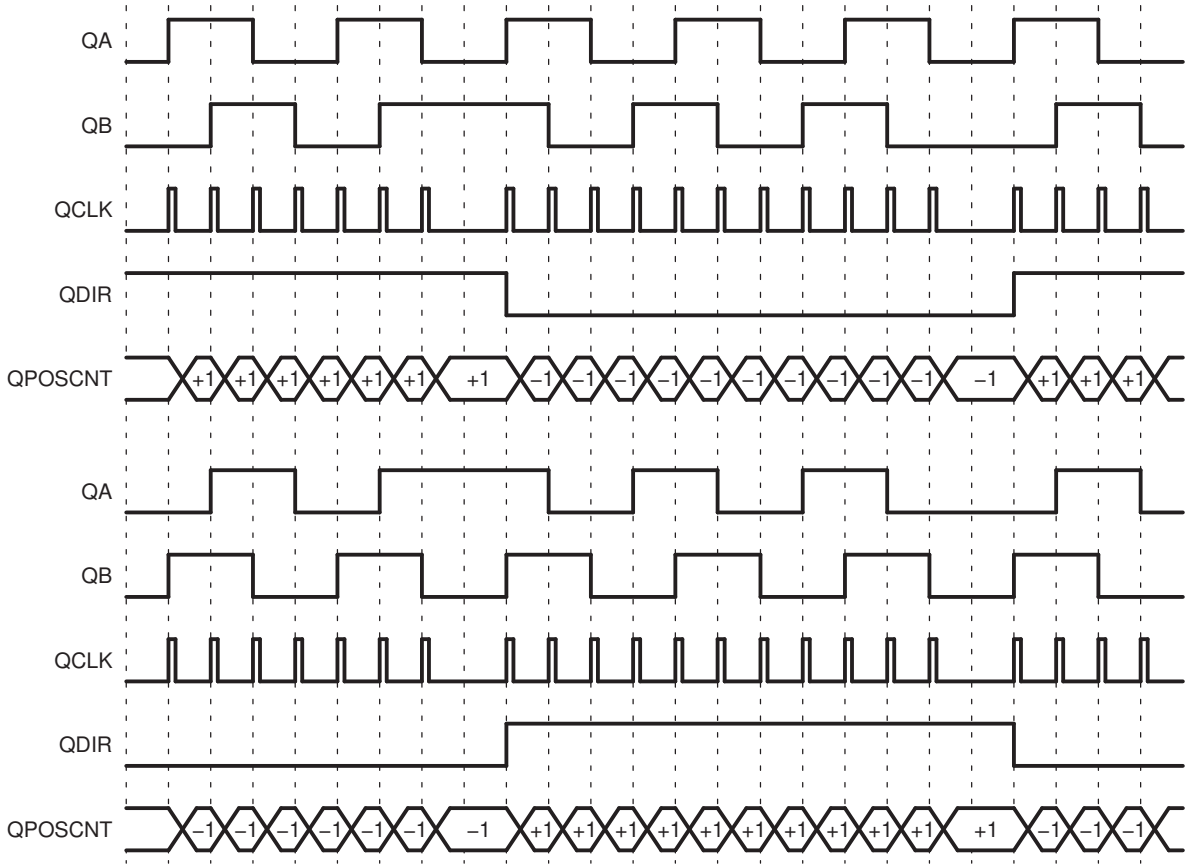
.			
Previous Edge	**Present Edge**	**QDIR**	**QPOSCNT**
QA↑	QB↑	UP	Increment
	QB↓	DOWN	Decrement
	QA↓	TOGGLE	Increment or Decrement
QA↓	QB↓	UP	Increment
	QB↑	DOWN	Decrement
	QA↑	TOGGLE	Increment or Decrement
QB↑	QA↑	DOWN	Increment
	QA↓	UP	Decrement
	QB↓	TOGGLE	Increment or Decrement
QB↓	QA↓	DOWN	Increment
	QA↑	UP	Decrement
	QB↑	TOGGLE	Increment or Decrement

Figure 9-6. Quadrature Decoder State Machine



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 9-7. Quadrature-clock and Direction Decoding



Phase Error Flag— In normal operating conditions, quadrature inputs QEPA and QEPB will be 90 degrees out of phase. The phase error flag (PHE) is set in the QFLG register when edge transition is detected simultaneously on the QEPA and QEPB signals to optionally generate interrupts. State transitions marked by dashed lines in Figure 9-6 are invalid transitions that generate a phase error.

Count Multiplication— The eQEP position counter provides 4x times the resolution of an input clock by generating a quadrature-clock (QCLK) on the rising/falling edges of both eQEP input clocks (QEPA and QEPB) as shown in Figure 9-7 .

Reverse Count— In normal quadrature count operation, QEPA input is fed to the QA input of the quadrature decoder and the QEPB input is fed to the QB input of the quadrature decoder. Reverse counting is enabled by setting the SWAP bit in the QDECCTL register. This will swap the input to the quadrature decoder thereby reversing the counting direction.

9.3.1.2 Direction-count Mode

Some position encoders provide direction and clock outputs, instead of quadrature outputs. In such cases, direction-count mode can be used. QEPA input will provide the clock for position counter and the QEPB input will have the direction information. The position counter is incremented on every rising edge of a QEPA input when the direction input is high and decremented when the direction input is low.

9.3.1.3 Up-Count Mode

The counter direction signal is hard-wired for up count and the position counter is used to measure the frequency of the QEPA input. Clearing the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of the QEPA input, thereby increasing the measurement resolution by 2x factor. In up-count mode, it is recommended that the application not configure QEPB as a GPIO mux option, or ensure that a signal edge is not generated on the QEPB input.

Copyright © 2012–2019, Texas Instruments Incorporated

9.3.1.4 Down-Count Mode

The counter direction signal is hardwired for a down count and the position counter is used to measure the frequency of the QEPA input. Setting of the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of a QEPA input, thereby increasing the measurement resolution by 2x factor. In down-count mode, it is recommended that the application not configure QEPB as a GPIO mux option, or ensure that a signal edge is not generated on the QEPB input.

9.3.2 eQEP Input Polarity Selection

Each eQEP input can be inverted using QDECCTL[8:5] control bits. As an example, setting of QDECCTL[QIP] bit will invert the index input.

9.3.3 Position-Compare Sync Output

The enhanced eQEP peripheral includes a position-compare unit that is used to generate the position-compare sync signal on compare match between the position counter register (QPOSCNT) and the position-compare register (QPOSCMP). This sync signal can be output using an index pin or strobe pin of the EQEP peripheral.

Setting the QDECCTL[SOEN] bit enables the position-compare sync output and the QDECCTL[SPSEL] bit selects either an eQEP index pin or an eQEP strobe pin.

9.4 Position Counter and Control Unit (PCCU)

The position counter and control unit provides two configuration registers (QEPCTL and QPOSCTL) for setting up position counter operational modes, position counter initialization/latch modes and position-compare logic for sync signal generation.

9.4.1 Position Counter Operating Modes

Position counter data may be captured in different manners. In some systems, the position counter is accumulated continuously for multiple revolutions and the position counter value provides the position information with respect to the known reference. An example of this is the quadrature encoder mounted on the motor controlling the print head in the printer. Here the position counter is reset by moving the print head to the home position and then position counter provides absolute position information with respect to home position.

In other systems, the position counter is reset on every revolution using index pulse and position counter provides rotor angle with respect to index pulse position.

Position counter can be configured to operate in following four modes

- Position Counter Reset on Index Event
- Position Counter Reset on Maximum Position
- Position Counter Reset on the first Index Event
- Position Counter Reset on Unit Time Out Event (Frequency Measurement)

In all the above operating modes, position counter is reset to 0 on overflow and to QPOSMAX register value on underflow. Overflow occurs when the position counter counts up after QPOSMAX value. Underflow occurs when position counter counts down after "0". Interrupt flag is set to indicate overflow/underflow in QFLG register.

9.4.1.1 Position Counter Reset on Index Event (QEPCTL[PCRM]=00)

If the index event occurs during the forward movement, then position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock.

First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers, it also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for index event reset operation.

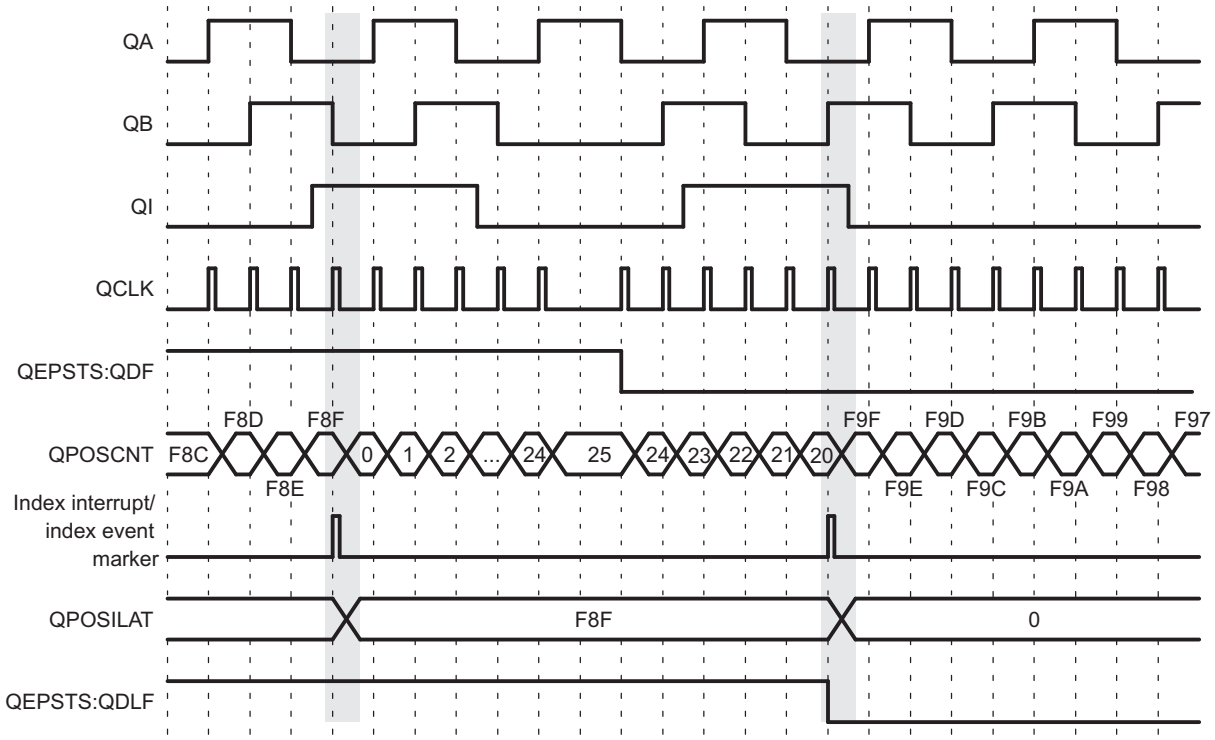
Copyright © 2012–2019, Texas Instruments Incorporated

For example, if the first reset operation occurs on the falling edge of QEPB during the forward direction, then all the subsequent reset must be aligned with the falling edge of QEPB for the forward rotation and on the rising edge of QEPB for the reverse rotation as shown in Figure 9-8.

The position-counter value is latched to the QPOSILAT register and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker. The position-counter error flag (QEPSTS[PCEF]) and error interrupt flag (QFLG[PCE]) are set if the latched value is not equal to 0 or QPOSMAX. The position-counter error flag (QEPSTS[PCEF]) is updated on every index event marker and an interrupt flag (QFLG[PCE]) will be set on error that can be cleared only through software.

The index event latch configuration QEPCTL[IEL] bits are ignored in this mode and position counter error flag/interrupt flag are generated only in index event reset mode.

Figure 9-8. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or 0xF9F)



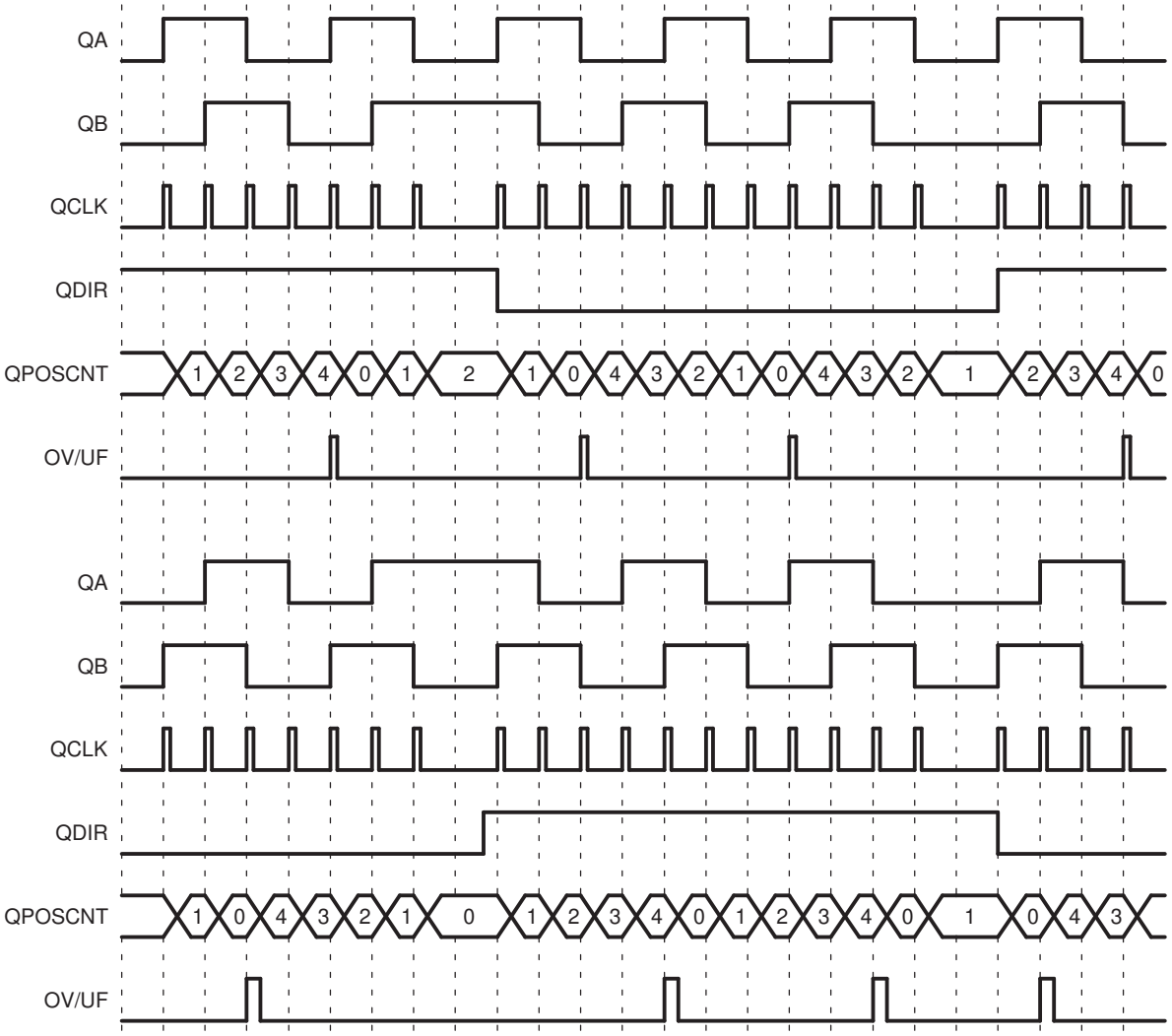
9.4.1.2 Position Counter Reset on Maximum Position (QEPCTL[PCRM]=01)

If the position counter is equal to QPOSMAX, then the position counter is reset to 0 on the next eQEP clock for forward movement and position counter overflow flag is set. If the position counter is equal to ZERO, then the position counter is reset to QPOSMAX on the next QEP clock for reverse movement and position counter underflow flag is set. Figure 9-9shows the position counter reset operation in this mode.

The first index marker fields (QEPSTS[FIDF] and QEPSTS[FIMF]) are not applicable in this mode.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 9-9. Position Counter Underflow/Overflow (QPOSMAX = 4)



9.4.1.3 Position Counter Reset on the First Index Event (QEPCTL[PCRM] = 10)

If the index event occurs during forward movement, then the position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock. Note that this is done only on the first occurrence and subsequently the position counter value is not reset on an index event; rather, it is reset based on maximum position as described in Section Section 9.4.1.2.

The first index marker fields (QEPSTS[FIDF] and QEPSTS[FIMF]) are not applicable in this mode.

9.4.1.4 Position Counter Reset on Unit Time out Event (QEPCTL[PCRM] = 11)

In this mode, the QPOSCNT value is latched to the QPOSLAT register and then the QPOSCNT is reset (to 0 or QPOSMAX, depending on the direction mode selected by QDECCTL[QSRC] bits on a unit time event). This is useful for frequency measurement.

9.4.2 Position Counter Latch

The eQEP index and strobe input can be configured to latch the position counter (QPOSCNT) into QPOSILAT and QPOSSLAT, respectively, on occurrence of a definite event on these pins.

Copyright © 2012–2019, Texas Instruments Incorporated

9.4.2.1 Index Event Latch

In some applications, it may not be desirable to reset the position counter on every index event and instead it may be required to operate the position counter in full 32-bit mode (QEPCTL[PCRM] = 01 and QEPCTL[PCRM] = 10 modes).

In such cases, the eQEP position counter can be configured to latch on the following events and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker.

- Latch on Rising edge (QEPCTL[IEL]=01)
- Latch on Falling edge (QEPCTL[IEL]=10)
- Latch on Index Event Marker (QEPCTL[IEL]=11)

This is particularly useful as an error checking mechanism to check if the position counter accumulated the correct number of counts between index events. As an example, the 1000-line encoder must count 4000 times when moving in the same direction between the index events.

The index event latch interrupt flag (QFLG[IEL]) is set when the position counter is latched to the QPOSILAT register.

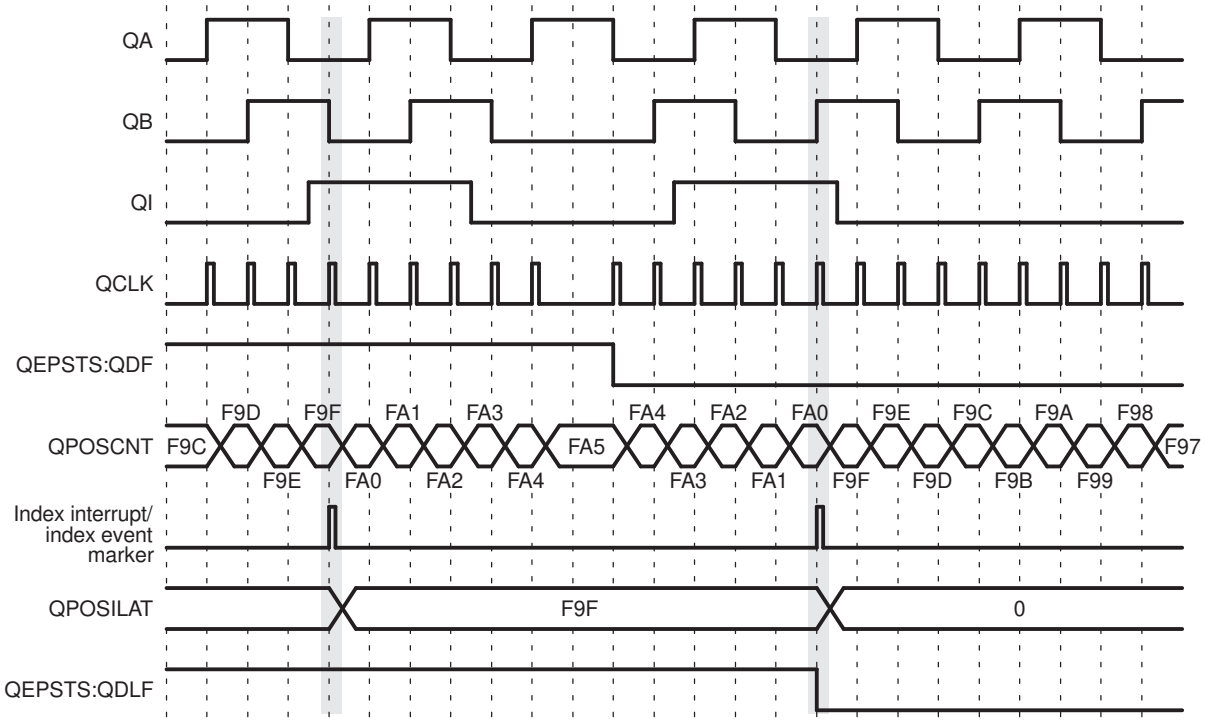
Latch on Rising Edge (QEPCTL[IEL]=01)— The position counter value (QPOSCNT) is latched to the QPOSILAT register on every rising edge of an index input.

Latch on Falling Edge (QEPCTL[IEL] = 10)— The position counter value (QPOSCNT) is latched to the QPOSILAT register on every falling edge of index input.

Latch on Index Event Marker/Software Index Marker (QEPCTL[IEL] = 11— The first index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in the QEPSTS registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for latching the position counter (QEPCTL[IEL]=11).

Figure 9-10 shows the position counter latch using an index event marker.

Figure 9-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1)



Copyright © 2012–2019, Texas Instruments Incorporated

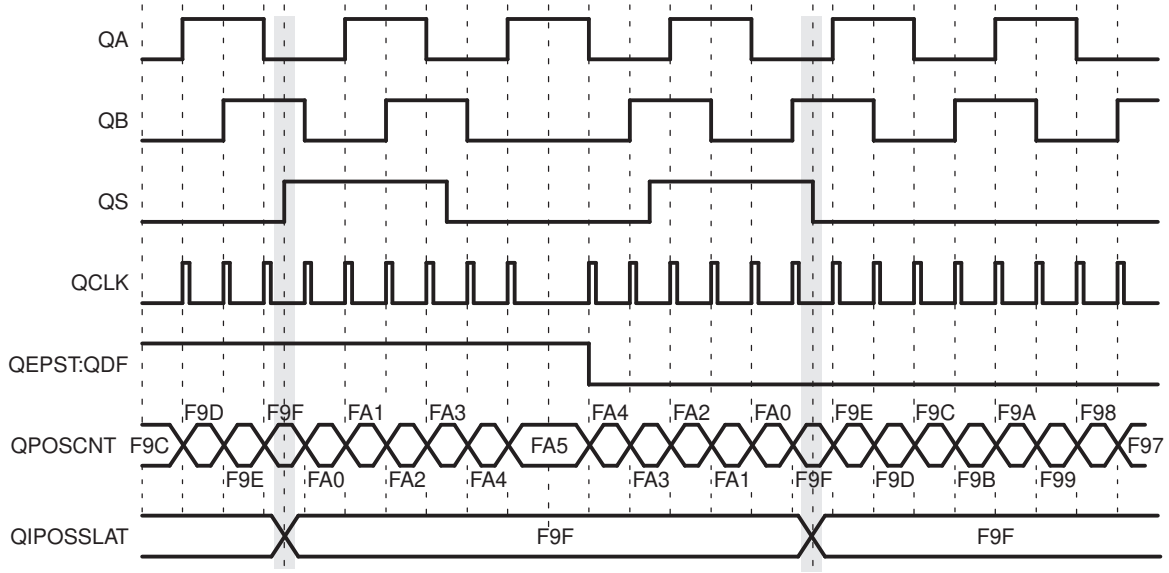
9.4.2.2 Strobe Event Latch

The position-counter value is latched to the QPOSSLAT register on the rising edge of the strobe input by clearing the QEPCTL[SEL] bit.

If the QEPCTL[SEL] bit is set, then the position counter value is latched to the QPOSSLAT register on the rising edge of the strobe input for forward direction and on the falling edge of the strobe input for reverse direction as shown in Figure 9-11.

The strobe event latch interrupt flag (QFLG[SEL) is set when the position counter is latched to the QPOSSLAT register.

Figure 9-11. Strobe Event Latch (QEPCTL[SEL] = 1)



9.4.3 Position Counter Initialization

The position counter can be initialized using following events:

- Index event
- Strobe event
- Software initialization

Index Event Initialization (IEI)— The QEPI index input can be used to trigger the initialization of the position counter at the rising or falling edge of the index input. If the QEPCTL[IEI] bits are 10, then the position counter (QPOSCNT) is initialized with a value in the QPOSINIT register on the rising edge of index input. Conversely, if the QEPCTL[IEI] bits are 11, initialization will be on the falling edge of the index input.

Strobe Event Initialization (SEI)— If the QEPCTL[SEI] bits are 10, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input.

If QEPCTL[SEL] bits are 11, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

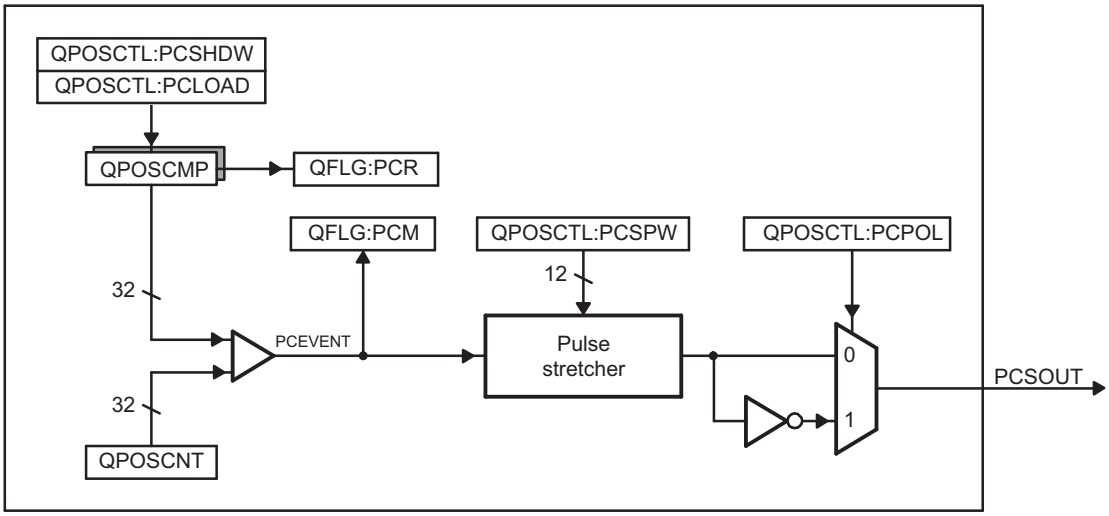
Software Initialization (SWI)— The position counter can be initialized in software by writing a 1 to the QEPCTL[SWI] bit. This bit is not automatically cleared. While the bit is still set, if a 1 is written to it again, the position counter will be re-initialized.

Copyright © 2012–2019, Texas Instruments Incorporated

9.4.4 eQEP Position-compare Unit

The eQEP peripheral includes a position-compare unit that is used to generate a sync output and/or interrupt on a position-compare match. Figure 9-12 shows a diagram. The position-compare (QPOSCMP) register is shadowed and shadow mode can be enabled or disabled using the QPOSCTL[PSSHDW] bit. If the shadow mode is not enabled, the CPU writes directly to the active position compare register.

Figure 9-12. eQEP Position-compare Unit



In shadow mode, you can configure the position-compare unit (QPOSCTL[PCLOAD]) to load the shadow register value into the active register on the following events and to generate the position-compare ready (QFLG[PCR]) interrupt after loading.

- Load on compare match
- Load on position-counter zero event

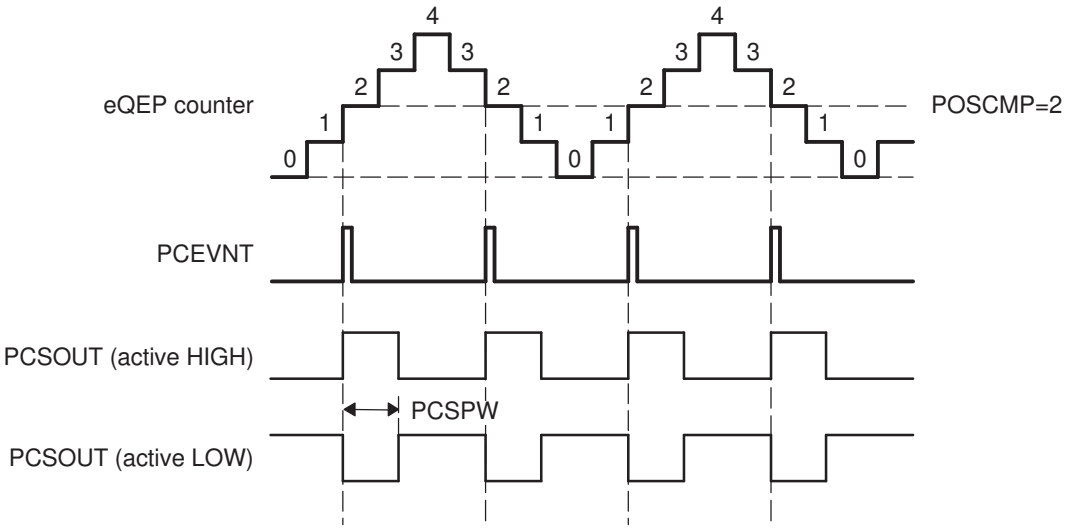
The position-compare match (QFLG[PCM]) is set when the position-counter value (QPOSCNT) matches with the active position-compare register (QPOSCMP) and the position-compare sync output of the programmable pulse width is generated on compare match to trigger an external device.

For example, if QPOSCMP = 2, the position-compare unit generates a position-compare event on 1 to 2 transitions of the eQEP position counter for forward counting direction and on 3 to 2 transitions of the eQEP position counter for reverse counting direction (see Figure 9-13).

Figure 9-23 shows the layout of the eQEP Position-Compare Control Register (QPOSCTL) and Table 9-5 describes the QPOSCTL bit fields.

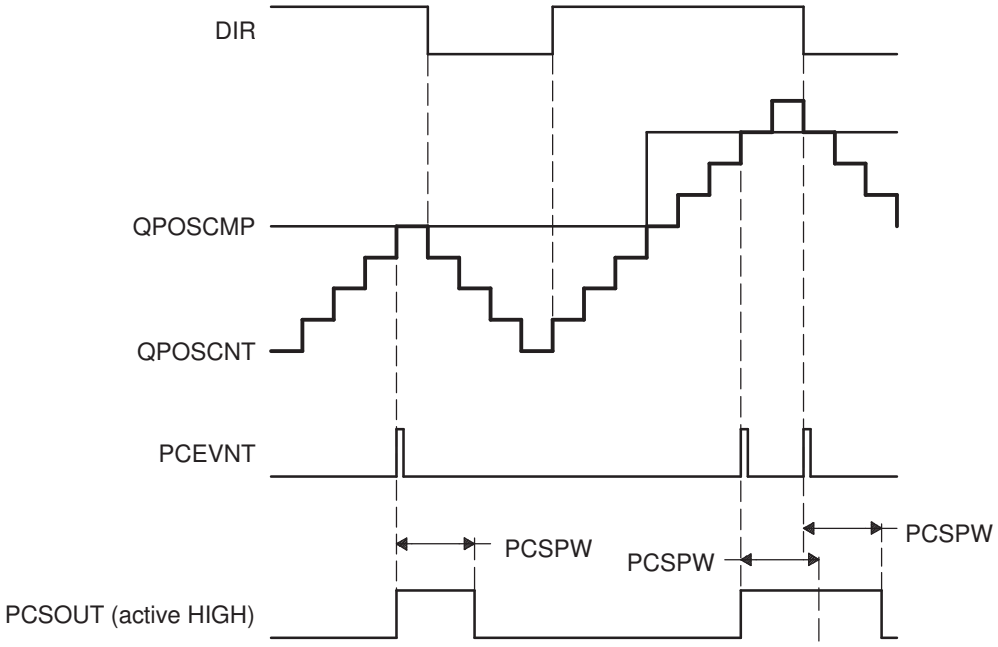
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 9-13. eQEP Position-compare Event Generation Points



The pulse stretcher logic in the position-compare unit generates a programmable position-compare sync pulse output on the position-compare match. In the event of a new position-compare match while a previous position-compare pulse is still active, then the pulse stretcher generates a pulse of specified duration from the new position-compare event as shown in Figure 9-14.

Figure 9-14. eQEP Position-compare Sync Output Pulse Stretcher



9.5 eQEP Edge Capture Unit

The eQEP peripheral includes an integrated edge capture unit to measure the elapsed time between the unit position events as shown in Figure 9-15. This feature is typically used for low speed measurement using the following equation:

$$v(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (3)$$

where,

Copyright © 2012–2019, Texas Instruments Incorporated

- X - Unit position is defined by integer multiple of quadrature edges (see Figure 9-16)
- ΔT - Elapsed time between unit position events
- v(k) - Velocity at time instant "k"

The eQEP capture timer (QCTMR) runs from prescaled SYSCLKOUT and the prescaler is programmed by the QCAPCTL[CCPS] bits. The capture timer (QCTMR) value is latched into the capture period register (QCPRD) on every unit position event and then the capture timer is reset, a flag is set in QEPSTS:UPEVNT to indicate that new value is latched into the QCPRD register. Software can check this status flag before reading the period register for low speed measurement and clear the flag by writing 1.

Time measurement (ΔT) between unit position events will be correct if the following conditions are met:

- No more than 65,535 counts have occurred between unit position events.
- No direction change between unit position events.

The capture unit sets the eQEP overflow error flag (QEPSTS[COEF]) in the event of capture timer overflow between unit position events. If a direction change occurs between the unit position events, then an error flag is set in the status register (QEPSTS[CDEF]).

Capture Timer (QCTMR) and Capture period register (QCPRD) can be configured to latch on following events.

- CPU read of QPOSCNT register
- Unit time-out event

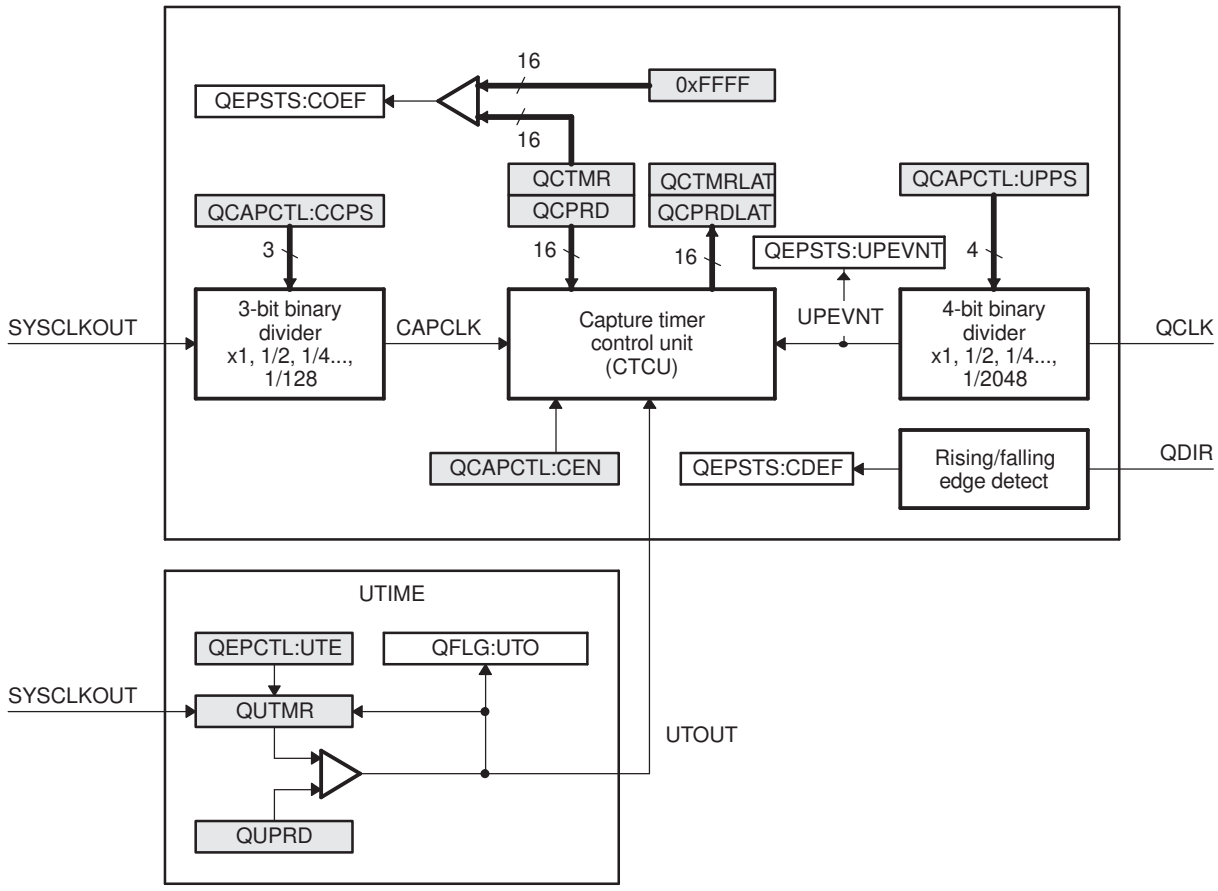
If the QEPCTL[QCLM] bit is cleared, then the capture timer and capture period values are latched into the QCTMRLAT and QCPRDLAT registers, respectively, when the CPU reads the position counter (QPOSCNT).

If the QEPCTL[QCLM] bit is set, then the position counter, capture timer, and capture period values are latched into the QPOSLAT, QCTMRLAT and QCPRDLAT registers, respectively, on unit time out.

Figure 9-17 shows the capture unit operation along with the position counter.

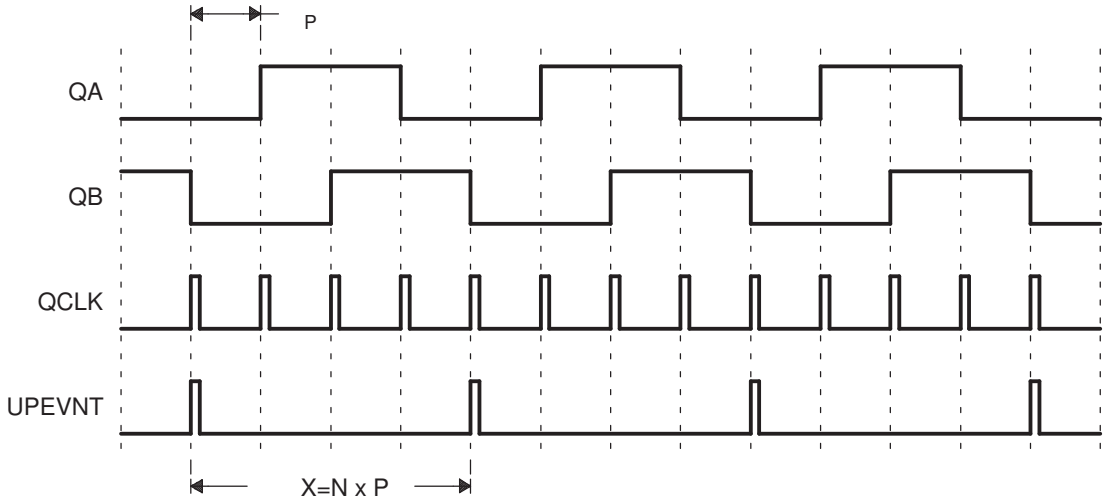
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 9-15. eQEP Edge Capture Unit



NOTE: The QCAPCTL[UPPS] prescaler should not be modified dynamically (such as switching the unit event prescaler from QCLK/4 to QCLK/8). Doing so may result in undefined behavior. The QCAPCTL[CPPS] prescaler can be modified dynamically (such as switching CAPCLK prescaling mode from SYSCLK/4 to SYSCLK/8) only after the capture unit is disabled.

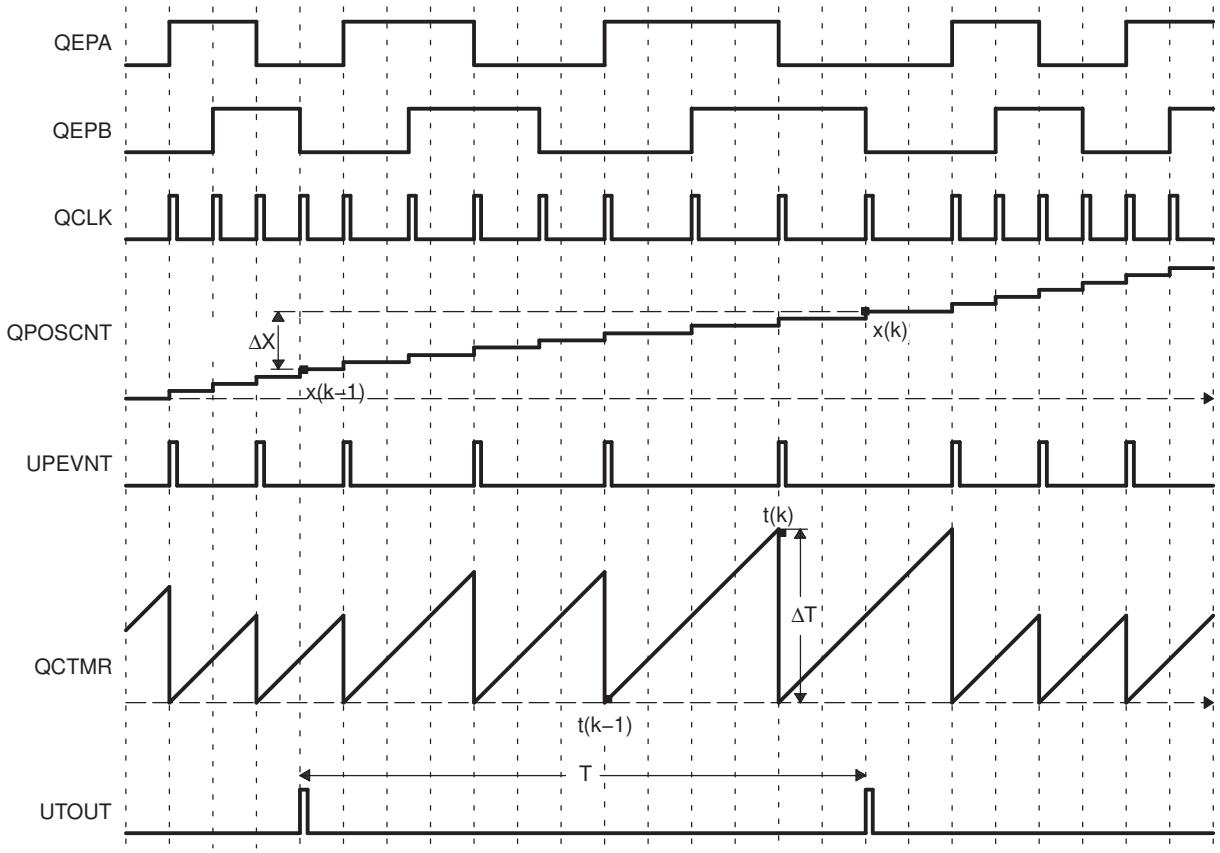
Figure 9-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010)



A N - Number of quadrature periods selected using QCAPCTL[UPPS] bits

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 9-17. eQEP Edge Capture Unit - Timing Details



Velocity Calculation Equations:

$$v(k) = \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (4)$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k-1

T: Fixed unit time or inverse of velocity calculation rate

ΔX: Incremental position movement in unit time

X: Fixed unit position

ΔT: Incremental time elapsed for unit position movement

t(k): Time instant "k"

t(k-1): Time instant "k-1"

Unit time (T) and unit period(X) are configured using the QUPRD and QCAPCTL[UPPS] registers. Incremental position output and incremental time output is available in the QPOSLAT and QCPRDLAT registers.

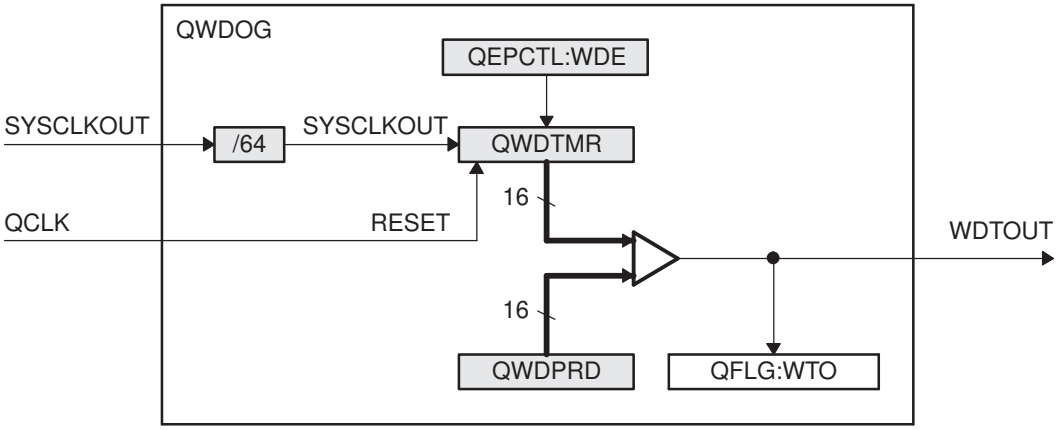
Copyright © 2012–2019, Texas Instruments Incorporated

Parameter	Relevant Register to Configure or Read the Information
T	Unit Period Register (QUPRD)
ΔX	Incremental Position = QPOSLAT(k) - QPOSLAT(K-1)
X	Fixed unit position defined by sensor resolution and ZCAPCTL[UPPS] bits
ΔT	Capture Period Latch (QCPRDLAT)

9.6 eQEP Watchdog

The eQEP peripheral contains a 16-bit watchdog timer that monitors the quadrature-clock to indicate proper operation of the motion-control system. The eQEP watchdog timer is clocked from SYSCLKOUT/64 and the quadrate clock event (pulse) resets the watchdog timer. If no quadrature-clock event is detected until a period match (QWDPRD = QWDTMR), then the watchdog timer will time out and the watchdog interrupt flag will be set (QFLG[WTO]). The time-out value is programmable through the watchdog period register (QWDPRD).

Figure 9-18. eQEP Watchdog Timer

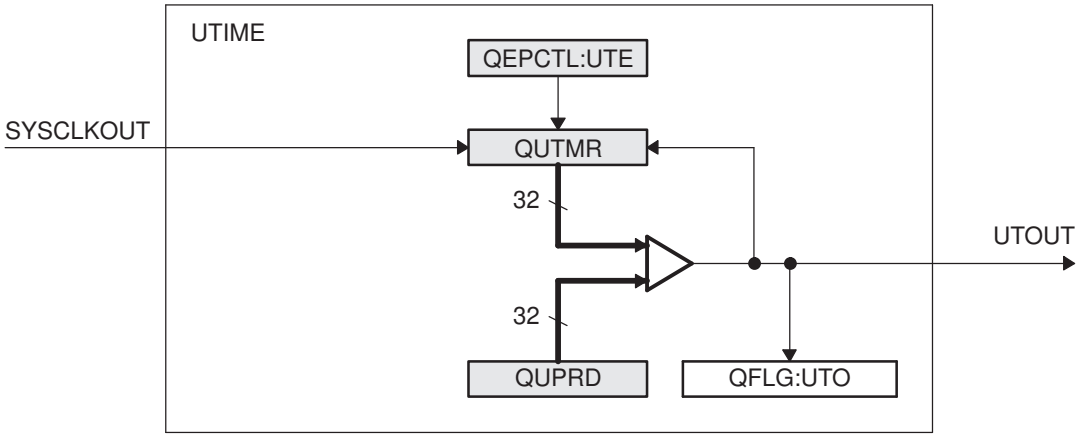


9.7 Unit Timer Base

The eQEP peripheral includes a 32-bit timer (QUTMR) that is clocked by SYSCLKOUT to generate periodic interrupts for velocity calculations. The unit time out interrupt is set (QFLG[UTO]) when the unit timer (QUTMR) matches the unit period register (QUPRD).

The eQEP peripheral can be configured to latch the position counter, capture timer, and capture period values on a unit time out event so that latched values are used for velocity calculation as described in Section Section 9.5.

Figure 9-19. eQEP Unit Time Base

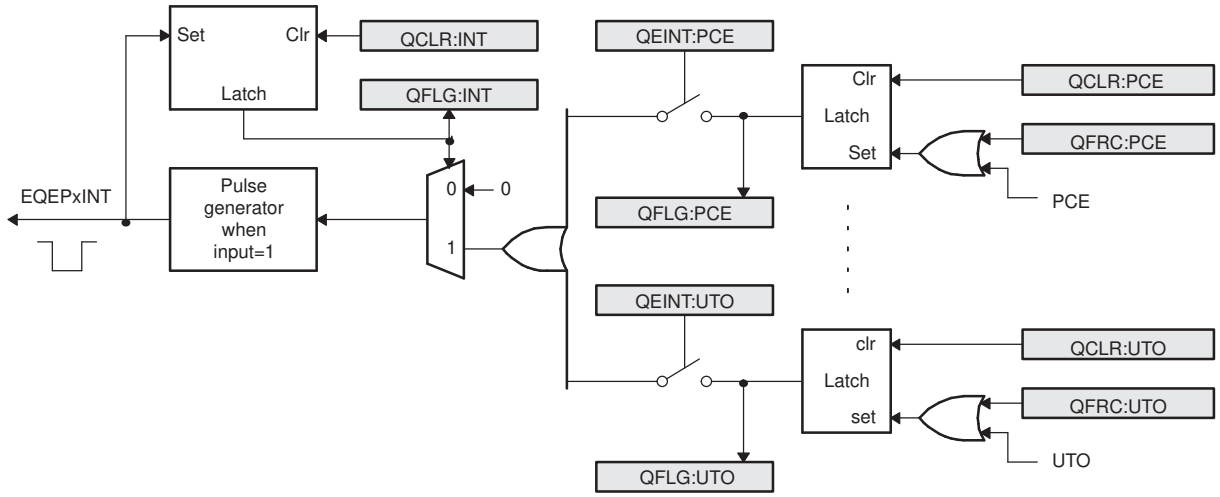


Copyright © 2012–2019, Texas Instruments Incorporated

9.8 eQEP Interrupt Structure

Figure 9-20 shows how the interrupt mechanism works in the EQEP module.

Figure 9-20. EQEP Interrupt Generation



Eleven interrupt events (PCE, PHE, QDC, WTO, PCU, PCO, PCR, PCM, SEL, IEL and UTO) can be generated. The interrupt control register (QEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (QFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated only to the PIE if any of the interrupt events is enabled, the flag bit is 1 and the INT flag bit is 0. The interrupt service routine will need to clear the global interrupt flag bit and the serviced event, via the interrupt clear register (QCLR), before any other interrupt pulses are generated. You can force an interrupt event by way of the interrupt force register (QFRC), which is useful for test purposes.

9.9 eQEP Registers

Figure 9-21. QEP Decoder Control (QDECCTL) Register

15	14	13	12	11	10	9	8
QSRC		SOEN	SPSEL	XCR	SWAP	IGATE	QAP
R/W-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	0
QBP	QIP	QSP	Reserved	
R/W-0	R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-3. eQEP Decoder Control (QDECCTL) Register Field Descriptions

Bits	Name	Value	Description
15-14	QSRC		Position-counter source selection
		00	Quadrature count mode (QCLK = iCLK, QDIR = iDIR)
		01	Direction-count mode (QCLK = xCLK, QDIR = xDIR)
		10	UP count mode for frequency measurement (QCLK = xCLK, QDIR = 1)
		11	DOWN count mode for frequency measurement (QCLK = xCLK, QDIR = 0)
13	SOEN		Sync output-enable
		0	Disable position-compare sync output
		1	Enable position-compare sync output

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-3. eQEP Decoder Control (QDECCTL) Register Field Descriptions (continued)

Bits	Name	Value	Description
12	SPSEL		Sync output pin selection
		0	Index pin is used for sync output
		1	Strobe pin is used for sync output
11	XCR		External clock rate
		0	2x resolution: Count the rising/falling edge
		1	1x resolution: Count the rising edge only
10	SWAP		Swap quadrature clock inputs. This swaps the input to the quadrature decoder, reversing the counting direction.
		0	Quadrature-clock inputs are not swapped
		1	Quadrature-clock inputs are swapped
9	IGATE		Index pulse gating option
		0	Disable gating of Index pulse
		1	Gate the index pin with strobe
8	QAP		QEPA input polarity
		0	No effect
		1	Negates QEPA input
7	QBP		QEPB input polarity
		0	No effect
		1	Negates QEPB input
6	QIP		QEPI input polarity
		0	No effect
		1	Negates QEPI input
5	QSP		QEPS input polarity
		0	No effect
		1	Negates QEPS input
4-0	Reserved		Always write as 0

Figure 9-22. eQEP Control (QEPCTL) Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREE, SOFT		PCRM		SEI		IEI		SWI	SEL	IEL		QPEN	QCLM	UTE	WDE
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	R/W-0	R/W-0		R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-4. eQEP Control (QEPCTL) Register Field Descriptions

Bits	Name	Value	Description
15-14	FREE, SOFT		Emulation Control Bits
			QPOSCNT behavior
		00	Position counter stops immediately on emulation suspend
		01	Position counter continues to count until the rollover
		1x	Position counter is unaffected by emulation suspend
			QWDTMR behavior
		00	Watchdog counter stops immediately
		01	Watchdog counter counts until WD period match roll over
		1x	Watchdog counter is unaffected by emulation suspend
			QUTMR behavior
		00	Unit timer stops immediately
		01	Unit timer counts until period rollover
		1x	Unit timer is unaffected by emulation suspend
			QCTMR behavior
		00	Capture Timer stops immediately
		01	Capture Timer counts until next unit period event
		1x	Capture Timer is unaffected by emulation suspend
13-12	PCRM		Position counter reset mode
		00	Position counter reset on an index event
		01	Position counter reset on the maximum position
		10	Position counter reset on the first index event
		11	Position counter reset on a unit time event
11-10	SEI		Strobe event initialization of position counter
		00	Does nothing (action disabled)
		01	Does nothing (action disabled)
		10	Initializes the position counter on rising edge of the QEPS signal
		11	Clockwise Direction: Initializes the position counter on the rising edge of QEPS strobe Counter Clockwise Direction: Initializes the position counter on the falling edge of QEPS strobe
9-8	IEI		Index event initialization of position counter
		00	Do nothing (action disabled)
		01	Do nothing (action disabled)
		10	Initializes the position counter on the rising edge of the QEPI signal (QPOSCNT = QPOSINIT)
		11	Initializes the position counter on the falling edge of QEPI signal (QPOSCNT = QPOSINIT)
7	SWI		Software initialization of position counter
		0	Do nothing (action disabled)
		1	Initialize position counter (QPOSCNT=QPOSINIT). This bit is not cleared automatically
6	SEL		Strobe event latch of position counter
		0	The position counter is latched on the rising edge of QEPS strobe (QPOSSLAT = POSCCNT). Latching on the falling edge can be done by inverting the strobe input using the QSP bit in the QDECCTL register.
		1	Clockwise Direction: Position counter is latched on rising edge of QEPS strobe Counter Clockwise Direction: Position counter is latched on falling edge of QEPS strobe
5-4	IEL		Index event latch of position counter (software index marker)
		00	Reserved
		01	Latches position counter on rising edge of the index signal
		10	Latches position counter on falling edge of the index signal
		11	Software index marker. Latches the position counter and quadrature direction flag on index event marker. The position counter is latched to the QPOSILAT register and the direction flag is latched in the QEPSTS[QDLF] bit. This mode is useful for software index marking.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-4. eQEP Control (QEPCTL) Register Field Descriptions (continued)

Bits	Name	Value	Description
3	QPEN		Quadrature position counter enable/software reset
		0	Reset the eQEP peripheral internal operating flags/read-only registers. Control/configuration registers are not disturbed by a software reset.
		1	eQEP position counter is enabled
2	QCLM		eQEP capture latch mode
		0	Latch on position counter read by CPU. Capture timer and capture period values are latched into QCTMRLAT and QCPRDLAT registers when CPU reads the QPOSCNT register.
		1	Latch on unit time out. Position counter, capture timer and capture period values are latched into QPOSLAT, QCTMRLAT and QCPRDLAT registers on unit time out.
1	UTE		eQEP unit timer enable
		0	Disable eQEP unit timer
		1	Enable unit timer
0	WDE		eQEP watchdog enable
		0	Disable the eQEP watchdog timer
		1	Enable the eQEP watchdog timer

Figure 9-23. eQEP Position-compare Control (QPOSCTL) Register

15	14	13	12	11	8
PCSHDW	PCLOAD	PCPOL	PCE	PCSPW	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

7	0
PCSPW	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-5. eQEP Position-compare Control (QPOSCTL) Register Field Descriptions

Bit	Name		Description
15	PCSHDW		Position-compare shadow enable
		0	Shadow disabled, load Immediate
		1	Shadow enabled
14	PCLOAD		Position-compare shadow load mode
		0	Load on QPOSCNT = 0
		1	Load when QPOSCNT = QPOSCMP
13	PCPOL		Polarity of sync output
		0	Active HIGH pulse output
		1	Active LOW pulse output
12	PCE		Position-compare enable/disable
		0	Disable position compare unit
		1	Enable position compare unit
11-0	PCSPW		Select-position-compare sync output pulse width
		0x000	1 * 4 * SYSCLKOUT cycles
		0x001	2 * 4 * SYSCLKOUT cycles
		0xFFF	4096 * 4 * SYSCLKOUT cycles

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 9-24. eQEP Capture Control (QCAPCTL) Register

15	14 7	6 4	3 0
CEN	Reserved	CCPS	UPPS
R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-6. eQEP Capture Control (QCAPCTL) Register Field Descriptions

Bits	Name	Description	
15	CEN		Enable eQEP capture
		0	eQEP capture unit is disabled
		1	eQEP capture unit is enabled
14-7	Reserved		Always write as 0
6-4	CCPS		eQEP capture timer clock prescaler
		000	CAPCLK = SYSCLKOUT/1
		001	CAPCLK = SYSCLKOUT/2
		010	CAPCLK = SYSCLKOUT/4
		011	CAPCLK = SYSCLKOUT/8
		100	CAPCLK = SYSCLKOUT/16
		101	CAPCLK = SYSCLKOUT/32
		110	CAPCLK = SYSCLKOUT/64
		111	CAPCLK = SYSCLKOUT/128
3-0	UPPS		Unit position event prescaler
		0000	UPEVNT = QCLK/1
		0001	UPEVNT = QCLK/2
		0010	UPEVNT = QCLK/4
		0011	UPEVNT = QCLK/8
		0100	UPEVNT = QCLK/16
		0101	UPEVNT = QCLK/32
		0110	UPEVNT = QCLK/64
		0111	UPEVNT = QCLK/128
		1000	UPEVNT = QCLK/256
		1001	UPEVNT = QCLK/512
		1010	UPEVNT = QCLK/1024
		1011	UPEVNT = QCLK/2048
		11xx	Reserved

Figure 9-25. eQEP Position Counter (QPOSCNT) Register

31 0
QPOSCNT
R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-7. eQEP Position Counter (QPOSCNT) Register Field Descriptions

Bits	Name	Description
31-0	QPOSCNT	This 32-bit position counter register counts up/down on every eQEP pulse based on direction input. This counter acts as a position integrator whose count value is proportional to position from a give reference point.

Figure 9-26. eQEP Position Counter Initialization (QPOSINIT) Register

31 0
QPOSINIT
R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-8. eQEP Position Counter Initialization (QPOSINIT) Register Field Descriptions

Bits	Name	Description
31-0	QPOSINIT	This register contains the position value that is used to initialize the position counter based on external strobe or index event. The position counter can be initialized through software. Writes to this register should always be full 32-bit writes.

Figure 9-27. eQEP Maximum Position Count Register (QPOSMAX) Register

31	0
QPOSMAX	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-9. eQEP Maximum Position Count (QPOSMAX) Register Field Descriptions

Bits	Name	Description
31-0	QPOSMAX	This register contains the maximum position counter value. Writes to this register should always be full 32-bit writes.

Figure 9-28. eQEP Position-compare (QPOSCMP) Register

31	0
QPOSCMP	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-10. eQEP Position-compare (QPOSCMP) Register Field Descriptions

Bits	Name	Description
31-0	QPOSCMP	The position-compare value in this register is compared with the position counter (QPOSCNT) to generate sync output and/or interrupt on compare match.

Figure 9-29. eQEP Index Position Latch (QPOSILAT) Register

31	0
QPOSILAT	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-11. eQEP Index Position Latch (QPOSILAT) Register Field Descriptions

Bits	Name	Description
31-0	QPOSILAT	The position-counter value is latched into this register on an index event as defined by the QEPCTL[IEL] bits.

Figure 9-30. eQEP Strobe Position Latch (QPOSSLAT) Register

31	0
QPOSSLAT	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-12. eQEP Strobe Position Latch (QPOSSLAT) Register Field Descriptions

Bits	Name	Description
31-0	QPOSSLAT	The position-counter value is latched into this register on strobe event as defined by the QEPCTL[SEL] bits.

Figure 9-31. eQEP Position Counter Latch (QPOSLAT) Register

31	0
QPOSLAT	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-13. eQEP Position Counter Latch (QPOSLAT) Register Field Descriptions

Bits	Name	Description
31-0	QPOSLAT	The position-counter value is latched into this register on unit time out event.

Figure 9-32. eQEP Unit Timer (QUTMR) Register

31	0
QUTMR	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-14. eQEP Unit Timer (QUTMR) Register Field Descriptions

Bits	Name	Description
31-0	QUTMR	This register acts as time base for unit time event generation. When this timer value matches with unit time period value, unit time event is generated.

Figure 9-33. eQEP Register Unit Period (QUPRD) Register

31	0
QUPRD	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-15. eQEP Unit Period (QUPRD) Register Field Descriptions

Bits	Name	Description
31-0	QUPRD	This register contains the period count for unit timer to generate periodic unit time events to latch the eQEP position information at periodic interval and optionally to generate interrupt. Writes to this register should always be full 32-bit writes.

Figure 9-34. eQEP Watchdog Timer (QWDTMR) Register

15	0
QWDTMR	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-16. eQEP Watchdog Timer (QWDTMR) Register Field Descriptions

Bits	Name	Description
15-0	QWDTMR	This register acts as time base for watch dog to detect motor stalls. When this timer value matches with watch dog period value, watch dog timeout interrupt is generated. This register is reset upon edge transition in quadrature-clock indicating the motion.

Figure 9-35. eQEP Watchdog Period (QWDPRD) Register

15	0
QWDPRD	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-17. eQEP Watchdog Period (QWDPRD) Register Field Description

Bits	Name	Value	Description
15-0	QWDPRD		This register contains the time-out count for the eQEP peripheral watch dog timer. When the watchdog timer value matches the watchdog period value, a watchdog timeout interrupt is generated.

Figure 9-36. eQEP Interrupt Enable (QEINT) Register

15	12	11	10	9	8
Reserved		UTO	IEL	SEL	PCM
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	QPE	PCE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-18. eQEP Interrupt Enable(QEINT) Register Field Descriptions

Bits	Name	Value	Description
15-12	Reserved	0	Always write as 0
11	UTO		Unit time out interrupt enable
		0	Interrupt is disabled
		1	Interrupt is enabled
10	IEL		Index event latch interrupt enable
		0	Interrupt is disabled
		1	Interrupt is enabled
9	SEL		Strobe event latch interrupt enable
		0	Interrupt is disabled
		1	Interrupt is enabled
8	PCM		Position-compare match interrupt enable
		0	Interrupt is disabled
		1	Interrupt is enabled
7	PCR		Position-compare ready interrupt enable
		0	Interrupt is disabled
		1	Interrupt is enabled
6	PCO		Position counter overflow interrupt enable

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-18. eQEP Interrupt Enable(QEINT) Register Field Descriptions (continued)

Bits	Name	Value	Description
		0	Interrupt is disabled
		1	Interrupt is enabled
5	PCU		Position counter underflow interrupt enable
		0	Interrupt is disabled
		1	Interrupt is enabled
4	WTO		Watchdog time out interrupt enable
		0	Interrupt is disabled
		1	Interrupt is enabled
3	QDC		Quadrature direction change interrupt enable
		0	Interrupt is disabled
		1	Interrupt is enabled
2	QPE		Quadrature phase error interrupt enable
		0	Interrupt is disabled
		1	Interrupt is enabled
1	PCE		Position counter error interrupt enable
		0	Interrupt is disabled
		1	Interrupt is enabled
0	Reserved		Reserved

Figure 9-37. eQEP Interrupt Flag (QFLG) Register

15	14	13	12	11	10	9	8
Reserved				UTO	IEL	SEL	PCM
R-0				R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	INT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-19. eQEP Interrupt Flag (QFLG) Register Field Descriptions

Bits	Name	Value	Description
15-12	Reserved		Always write as 0
11	UTO		Unit time out interrupt flag
		0	No interrupt generated
		1	Set by eQEP unit timer period match
10	IEL		Index event latch interrupt flag
		0	No interrupt generated
		1	This bit is set after latching the QPOSCNT to QPOSILAT
9	SEL		Strobe event latch interrupt flag
		0	No interrupt generated
		1	This bit is set after latching the QPOSCNT to QPOSSLAT
8	PCM		eQEP compare match event interrupt flag
		0	No interrupt generated
		1	This bit is set on position-compare match
7	PCR		Position-compare ready interrupt flag
		0	No interrupt generated
		1	This bit is set after transferring the shadow register value to the active position compare register.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-19. eQEP Interrupt Flag (QFLG) Register Field Descriptions (continued)

Bits	Name	Value	Description
6	PCO		Position counter overflow interrupt flag
		0	No interrupt generated
		1	This bit is set on position counter overflow.
5	PCU		Position counter underflow interrupt flag
		0	No interrupt generated
		1	This bit is set on position counter underflow.
4	WTO		Watchdog timeout interrupt flag
		0	No interrupt generated
		1	Set by watch dog timeout
3	QDC		Quadrature direction change interrupt flag
		0	No interrupt generated
		1	This bit is set during change of direction
2	PHE		Quadrature phase error interrupt flag
		0	No interrupt generated
		1	Set on simultaneous transition of QEPA and QEPB
1	PCE		Position counter error interrupt flag
		0	No interrupt generated
		1	Position counter error
0	INT		Global interrupt status flag
		0	No interrupt generated
		1	Interrupt was generated

Figure 9-38. eQEP Interrupt Clear (QCLR) Register

15			12	11	10	9	8
Reserved				UTO	IEL	SEL	PCM
R-0				R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	INT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-20. eQEP Interrupt Clear (QCLR) Register Field Descriptions

Bit	Field	Value	Description
15-12	Reserved		Always write as 0s
11	UTO		Clear unit time out interrupt flag
		0	No effect
		1	Clears the interrupt flag
10	IEL		Clear index event latch interrupt flag
		0	No effect
		1	Clears the interrupt flag
9	SEL		Clear strobe event latch interrupt flag
		0	No effect
		1	Clears the interrupt flag
8	PCM		Clear eQEP compare match event interrupt flag
		0	No effect
		1	Clears the interrupt flag

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-20. eQEP Interrupt Clear (QCLR) Register Field Descriptions (continued)

Bit	Field	Value	Description
7	PCR		Clear position-compare ready interrupt flag
		0	No effect
		1	Clears the interrupt flag
6	PCO		Clear position counter overflow interrupt flag
		0	No effect
		1	Clears the interrupt flag
5	PCU		Clear position counter underflow interrupt flag
		0	No effect
		1	Clears the interrupt flag
4	WTO		Clear watchdog timeout interrupt flag
		0	No effect
		1	Clears the interrupt flag
3	QDC		Clear quadrature direction change interrupt flag
		0	No effect
		1	Clears the interrupt flag
2	PHE		Clear quadrature phase error interrupt flag
		0	No effect
		1	Clears the interrupt flag
1	PCE		Clear position counter error interrupt flag
		0	No effect
		1	Clears the interrupt flag
0	INT		Global interrupt clear flag
		0	No effect
		1	Clears the interrupt flag and enables further interrupts to be generated if an event flags is set to 1.

Figure 9-39. eQEP Interrupt Force (QFRC) Register

15			12	11	10	9	8
Reserved				UTO	IEL	SEL	PCM
R-0				R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-21. eQEP Interrupt Force (QFRC) Register Field Descriptions

Bit	Field	Value	Description
15-12	Reserved		Always write as 0s
11	UTO		Force unit time out interrupt
		0	No effect
		1	Force the interrupt
10	IEL		Force index event latch interrupt
		0	No effect
		1	Force the interrupt
9	SEL		Force strobe event latch interrupt
		0	No effect
		1	Force the interrupt

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-21. eQEP Interrupt Force (QFRC) Register Field Descriptions (continued)

Bit	Field	Value	Description
8	PCM		Force position-compare match interrupt
		0	No effect
		1	Force the interrupt
7	PCR		Force position-compare ready interrupt
		0	No effect
		1	Force the interrupt
6	PCO		Force position counter overflow interrupt
		0	No effect
		1	Force the interrupt
5	PCU		Force position counter underflow interrupt
		0	No effect
		1	Force the interrupt
4	WTO		Force watchdog time out interrupt
		0	No effect
		1	Force the interrupt
3	QDC		Force quadrature direction change interrupt
		0	No effect
		1	Force the interrupt
2	PHE		Force quadrature phase error interrupt
		0	No effect
		1	Force the interrupt
1	PCE		Force position counter error interrupt
		0	No effect
		1	Force the interrupt
0	Reserved	0 1	Always write as 0

Figure 9-40. eQEP Status (QEPSTS) Register

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
UPEVNT	FIDF	QDF	QDLF	COEF	CDEF	FIMF	PCEF
R/W-1	R-0	R-0	R-0	R/W-1	R/W-1	R/W-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-22. eQEP Status (QEPSTS) Register Field Descriptions

Bit	Field	Value	Description
15-8	Reserved		Always write as 0
7	UPEVNT		Unit position event flag
		0	No unit position event detected
		1	Unit position event detected. Write 1 to clear.
6	FIDF		Direction on the first index marker Status of the direction is latched on the first index event marker.
		0	Counter-clockwise rotation (or reverse movement) on the first index event
		1	Clockwise rotation (or forward movement) on the first index event

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-22. eQEP Status (QEPSTS) Register Field Descriptions (continued)

Bit	Field	Value	Description
5	QDF		Quadrature direction flag
		0	Counter-clockwise rotation (or reverse movement)
		1	Clockwise rotation (or forward movement)
4	QDLF		eQEP direction latch flag Status of direction is latched on every index event marker.
		0	Counter-clockwise rotation (or reverse movement) on index event marker
		1	Clockwise rotation (or forward movement) on index event marker
3	COEF		Capture overflow error flag
		0	Sticky bit, cleared by writing 1
		1	Overflow occurred in eQEP Capture timer (QEPCTMR)
2	CDEF		Capture direction error flag
		0	Sticky bit, cleared by writing 1
		1	Direction change occurred between the capture position event.
1	FIMF		First index marker flag
		0	Sticky bit, cleared by writing 1
		1	Set by first occurrence of index pulse
0	PCEF		Position counter error flag. This bit is not sticky and it is updated for every index event.
		0	No error occurred during the last index transition.
		1	Position counter error

Figure 9-41. eQEP Capture Timer (QCTMR) Register

15	0
QCTMR	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-23. eQEP Capture Timer (QCTMR) Register Field Descriptions

Bits	Name	Description
15-0	QCTMR	This register provides time base for edge capture unit.

Figure 9-42. eQEP Capture Period (QCPRD) Register

15	0
QCPRD	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-24. eQEP Capture Period Register (QCPRD) Register Field Descriptions

Bits	Name	Description
15-0	QCPRD	This register holds the period count value between the last successive eQEP position events

Figure 9-43. eQEP Capture Timer Latch (QCTMRLAT) Register

15	0
QCTMRLAT	
R	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 9-25. eQEP Capture Timer Latch (QCTMRLAT) Register Field Descriptions

Bits	Name	Description
15-0	QCTMRLAT	The eQEP capture timer value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter.

Figure 9-44. eQEP Capture Period Latch (QCPRDLAT) Register

15	0
QCPRDLAT	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-26. eQEP Capture Period Latch (QCPRDLAT) Register Field Descriptions

Bits	Name	Description
15-0	QCPRDLAT	eQEP capture period value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter.

Copyright © 2012–2019, Texas Instruments Incorporated

Analog Subsystem

The ADC module described in this chapter is a Type 3 ADC and the Comparator function described here is a Type 0 Comparator. See the *TMS320C28xx, 28xxx DSP Peripheral Reference Guide* (SPRU566) for a list of all devices with modules of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

Topic **Page**

10.1 Overview **855**
10.2 Analog Common Interface Bus (ACIB) **855**
10.3 Analog-to-Digital Converter (ADC) **859**
10.4 Comparator Block **901**
10.5 Analog Subsystem Software **906**

Copyright © 2012–2019, Texas Instruments Incorporated

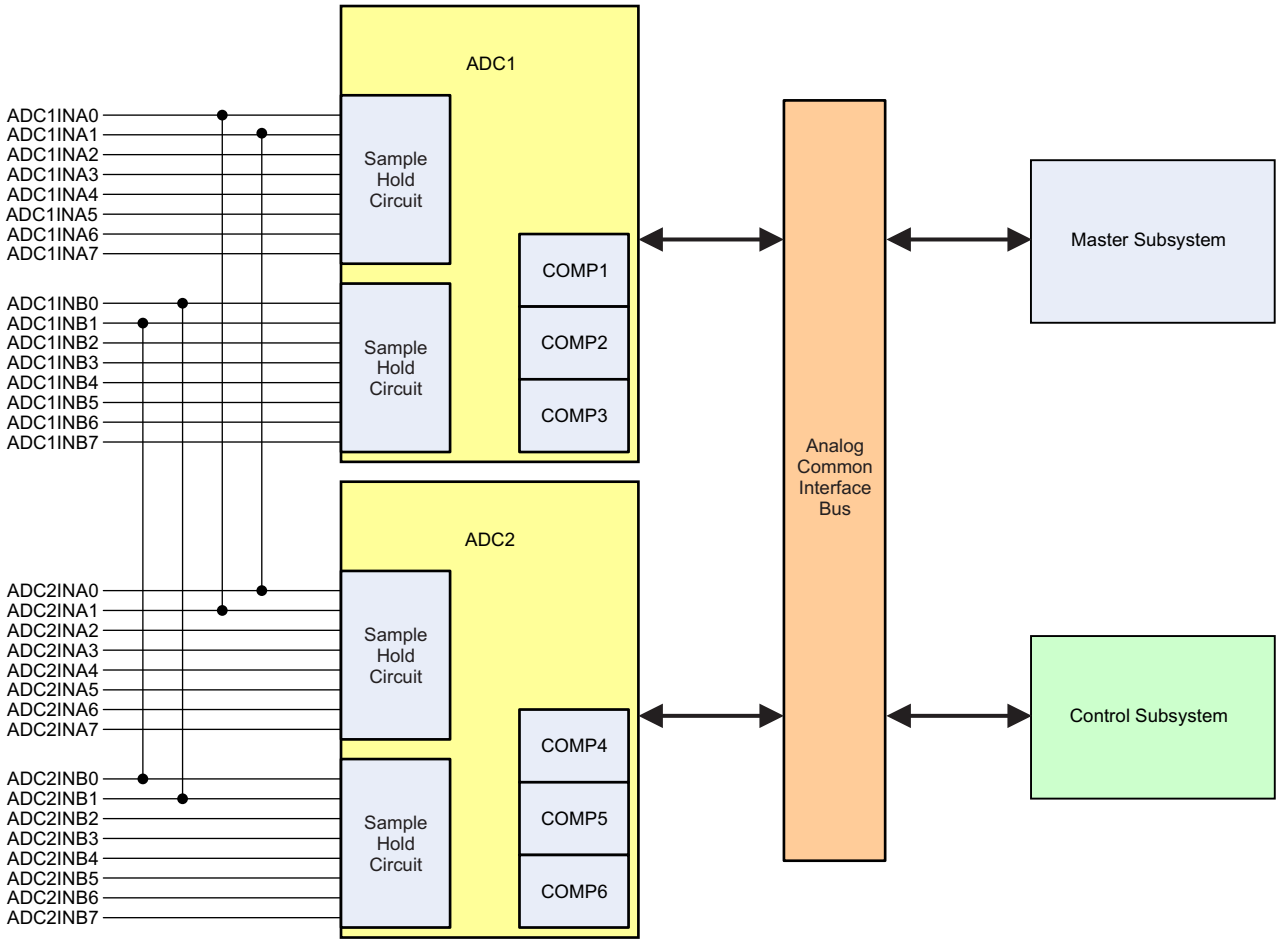
10.1 Overview

The analog subsystem consists of dual 12-bit ADC modules and six comparators with six internal 10-bit DACs. Each ADC module block has two sample and hold circuits and three comparators plus three internal 10-bit DACs.

The analog subsystem is accessed by the analog common interface bus (ACIB). The ACIB is responsible for transferring triggers initiated by the control subsystem to the analog subsystem and transferring interrupts initiated by the analog subsystem to both the master subsystem and control subsystem. The ACIB also transfers analog register read and write data and ADC conversion results.

Figure 10-1 shows the block diagram of the analog subsystem.

Figure 10-1. Analog Subsystem Block Diagram



Notice in the figure that ADC2 channel A1 is internally connected to ADC1 channel A0, and ADC1 channel A1 is internally connected to ADC2 channel A0. ADC2 channel B1 is internally connected to ADC1 channel B0, and ADC1 channel B1 is internally connected to ADC2 channel B0.

10.2 Analog Common Interface Bus (ACIB)

The ACIB handles all interaction between the analog subsystem and the digital subsystem (where digital subsystem includes the master and control subsystems). A simplified model of the ACIB is shown in Figure 10-2 and the signals are defined in Table 10-1.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-2. Simplified ACIB Model

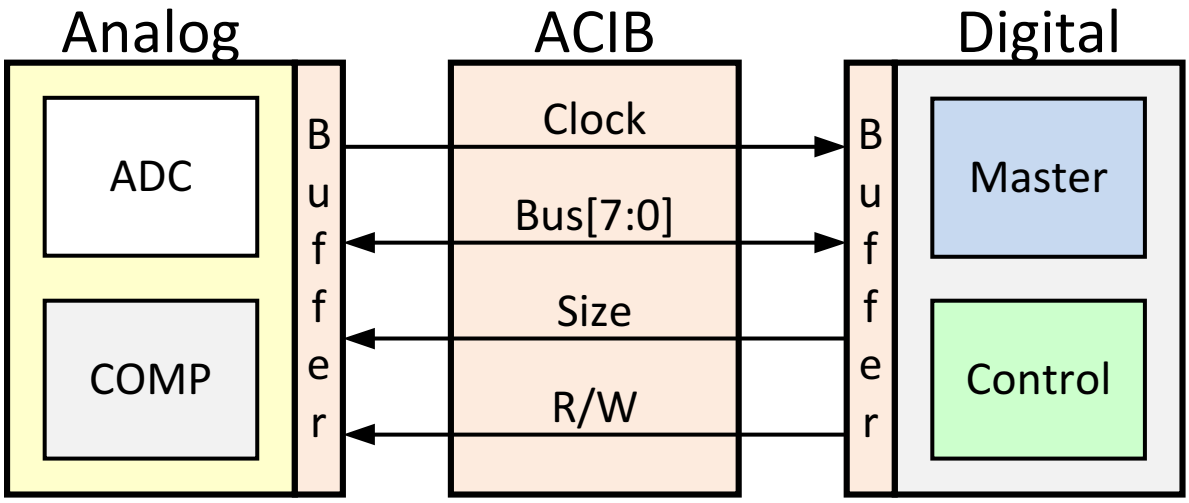


Table 10-1. Simplified ACIB Signals

Signal	Direction	Description
Clock	Analog to Digital	ACIB clock
Bus[7:0]	Bidirectional	General purpose bus that carries address, data, trigger, and interrupt information.
Size	Digital to Analog	Size of the requested read or write operation. The Size signal can only request 16-bit or 32-bit data operations. A burst read of 64-bits is automatically initiated by requesting a 32-bit read on an address that is aligned to a 64-bit boundary -- there is no way to disable burst reads. Burst writes are not supported.
R/W	Digital to Analog	Type of data operation. Read or write, with respect to the Digital subsystem.

All ACIB activity operates at a frequency equal to the analog subsystem clock rate. Register accesses across the ACIB are handled as data read and write operations. Trigger and interrupt signals are specially encoded by the ACIB for lower latency behavior.

ACIB buffers on both the digital and analog subsystems act as an intermediary translation stage for any information that is transmitted and received across the bus. This translation stage will introduce latency beyond the time required for bus activity alone.

The translation latency is represented by the digital buffer and analog buffer signals in Figure 10-3 to Figure 10-9. The digital and analog subsystems are not able to use the information received across the ACIB bus until the respective buffer signal has de-asserted.

While the buffers are busy, a synchronization stall is active whenever the ACIB bus is not carrying data. For communications originating from the digital subsystem, there can be one cycle of jitter during the opening synchronization sequence.

Read and write operations are processed in the order in which they are received, but they will be preempted in place by ADC trigger and interrupt signals in case of contention. For instances where operations are pending from multiple CPU or DMA sources, the transactions are processed in a round-robin manner.

Copyright © 2012–2019, Texas Instruments Incorporated

The ACIB buffer can store a single write operation from the digital subsystem in order to prevent the CPU or DMA from stalling on writes. If a second write operation is initiated by the digital subsystem (before the first write operation completes), the CPU or DMA will stall until the first write operation completes, thereby freeing up the buffer for the second write operation.

Simplified timing diagrams for ACIB operations are shown in Figure 10-3 to Figure 10-9. The diagrams are approximations and may not be cycle accurate.

Figure 10-3. 16-bit Write

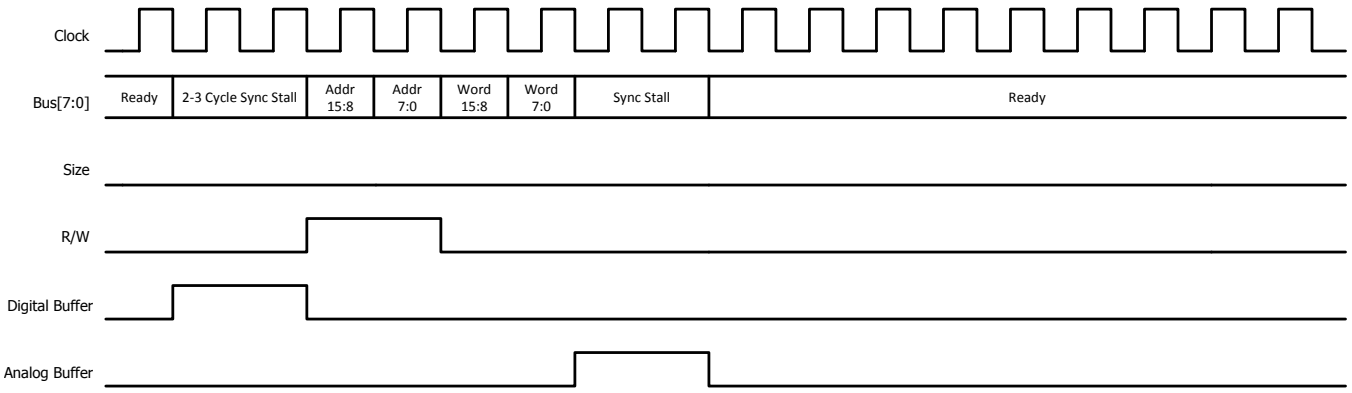


Figure 10-4. 32-bit Write

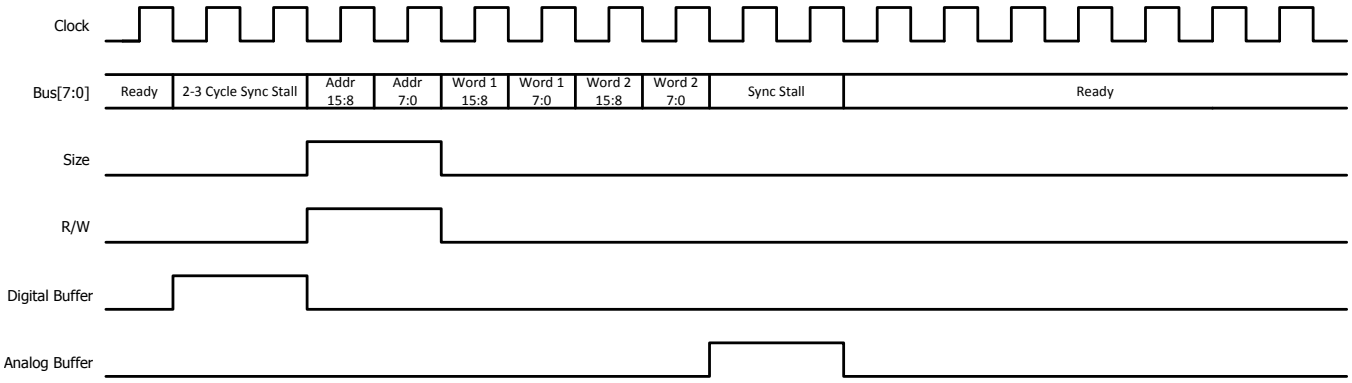
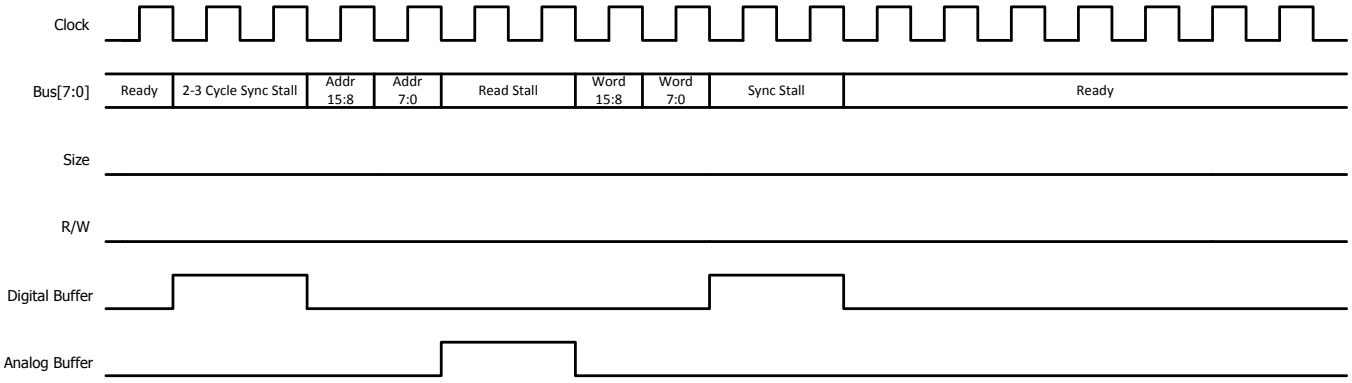


Figure 10-5. 16-bit Read



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-6. 32-bit Read

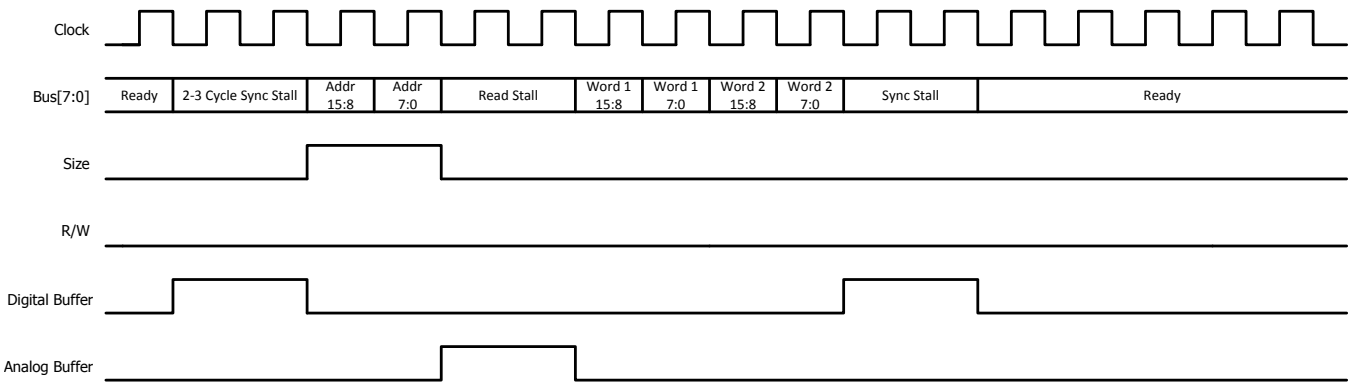


Figure 10-7. 64-bit Read

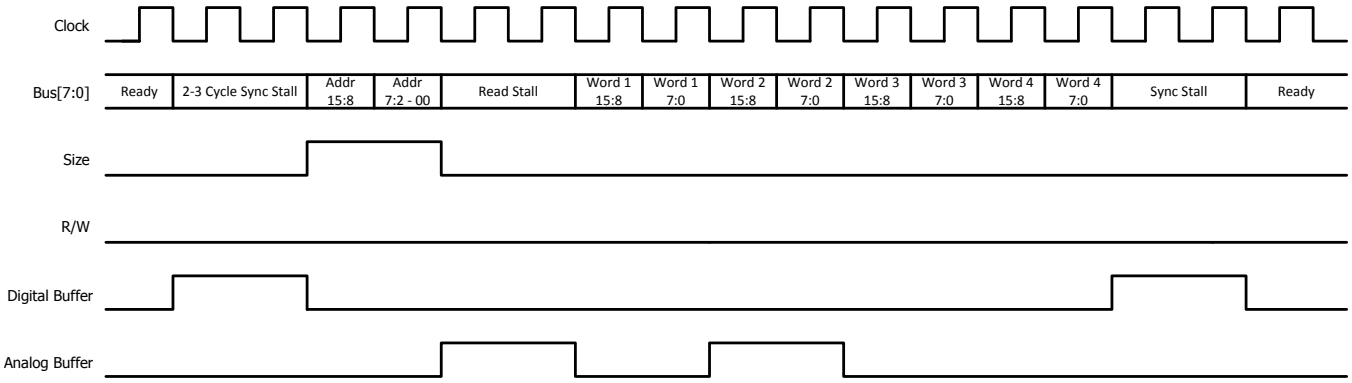
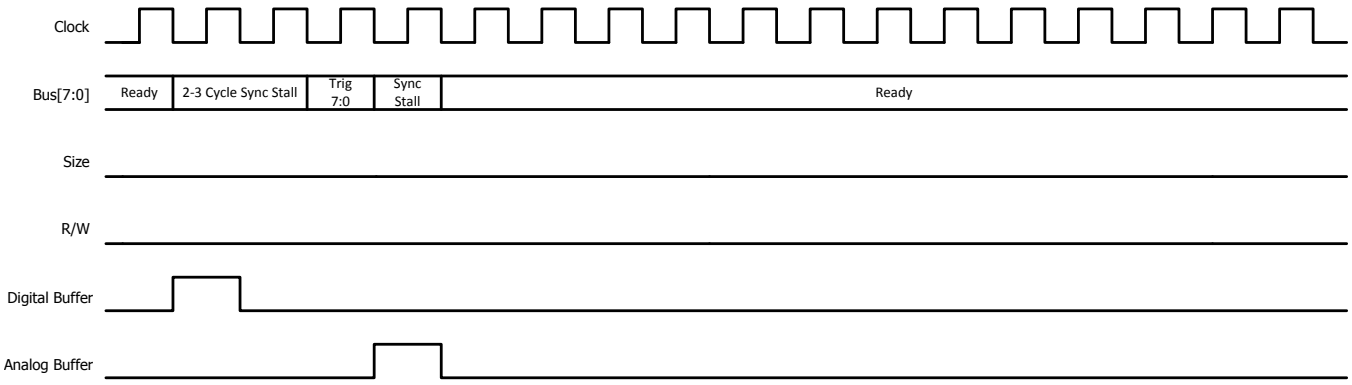
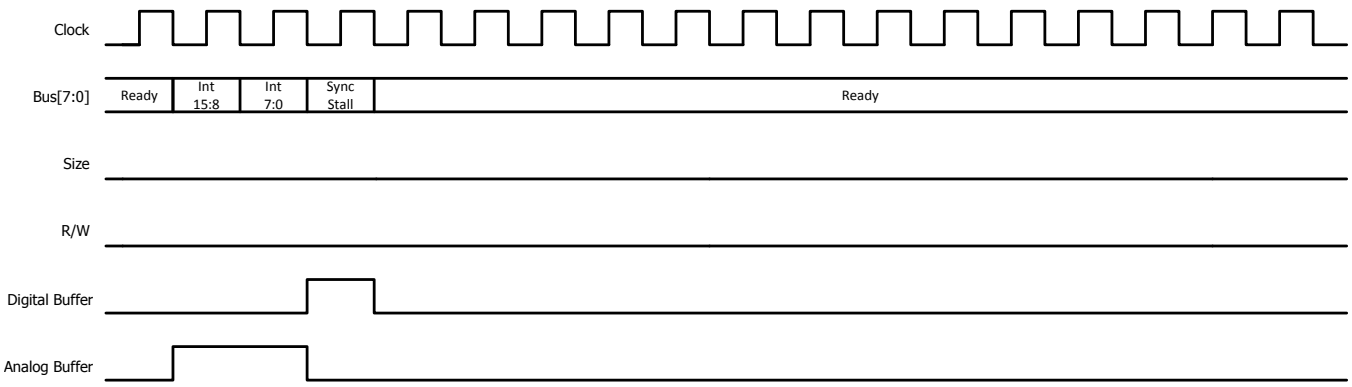


Figure 10-8. ADC Trigger



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-9. ADC Interrupt



10.3 Analog-to-Digital Converter (ADC)

The ADC module described in this reference guide is a 12-bit recyclic ADC; part SAR, part pipelined. The analog circuits of this converter, referred to as the "core" in this document, include the front-end analog multiplexers (MUXs), sample-and-hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the "wrapper" in this document, include programmable conversions, result registers, interface to analog circuits, interface to the Analog Common Interface Bus (ACIB), and interface to other on-chip modules.

10.3.1 Features

The core of the ADC contains a single 12-bit converter fed by two sample and hold circuits. The sample and hold circuits can be sampled simultaneously or sequentially. These, in turn, are fed by a total of up to 16 analog input channels. See the device datasheet for the specific number of channels available. The converter can be configured to run with an internal bandgap reference to create true-voltage based conversions or with a pair of external voltage references (VREFHI/LO) to create ratiometric based conversions.

Contrary to previous ADC types, this ADC is not sequencer based. It is easy for the user to create a series of conversions from a single trigger. However, the basic principle of operation is centered around the configurations of individual conversions, called SOC's, or start-of-conversions.

Functions of each ADC module include:

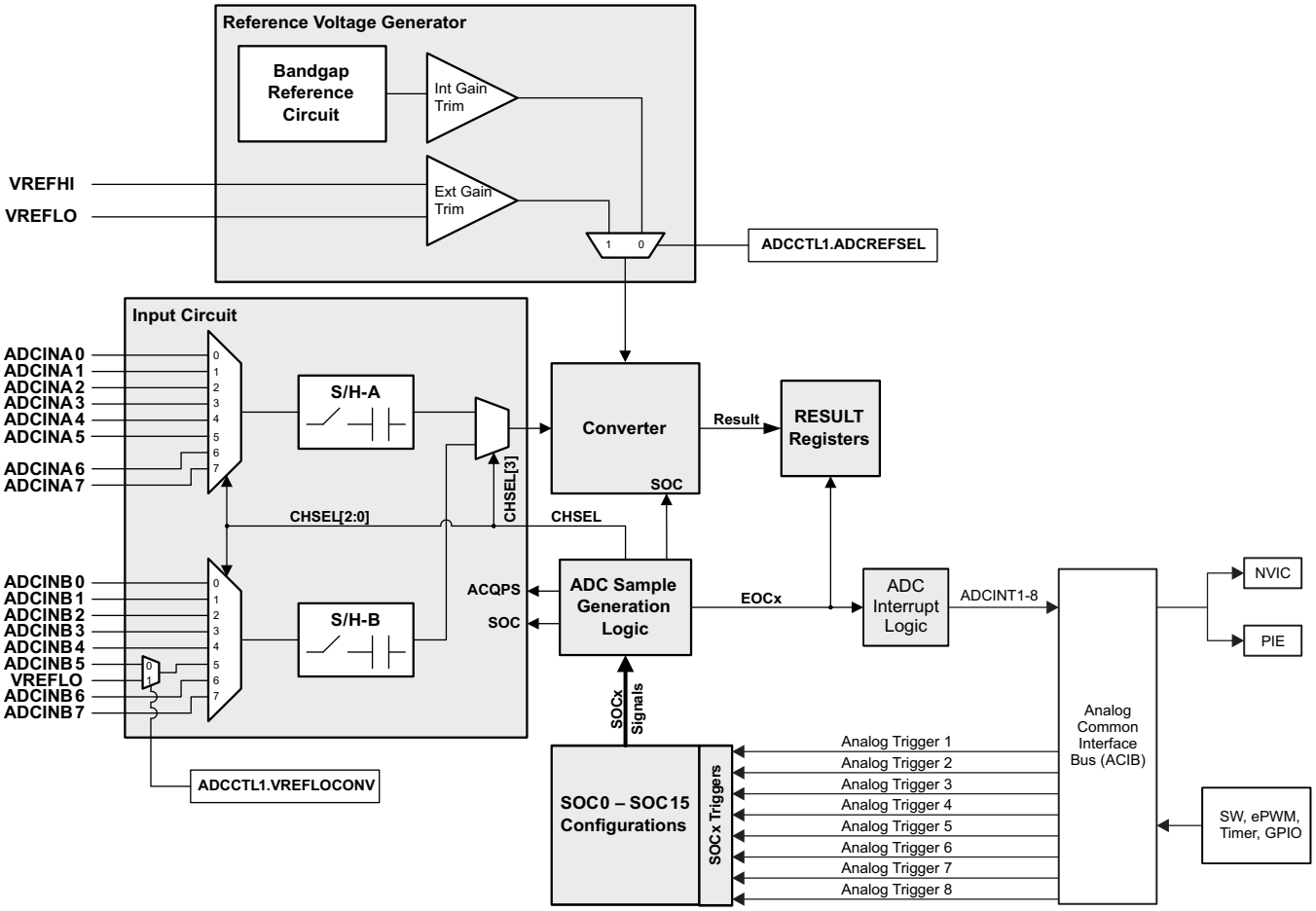
- 12-bit ADC core with built-in dual sample-and-hold (S/H)
- Simultaneous sampling or sequential sampling modes
- Full range analog input: 0 V to 3.3 V fixed, or VREFHI/VREFLO ratiometric
- Up to 16-channel, multiplexed inputs
- 16 SOC's, configurable for trigger, sample window, and channel
- 16 result registers (individually addressable) to store conversion values
- Multiple trigger sources
  - S/W - software immediate start
  - ePWM 1-9
  - External GPIO
  - CPU Timers 0/1/2
  - ADCINT1/2
- 8 flexible PIE interrupts total for both ADC modules, can configure interrupt request after any conversion
- 8 flexible NVIC interrupts total for both ADC modules, can configure interrupt request after any conversion

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.2 Block Diagram

Figure 10-10 shows the block diagram of the ADC module.

Figure 10-10. ADC Module Block Diagram



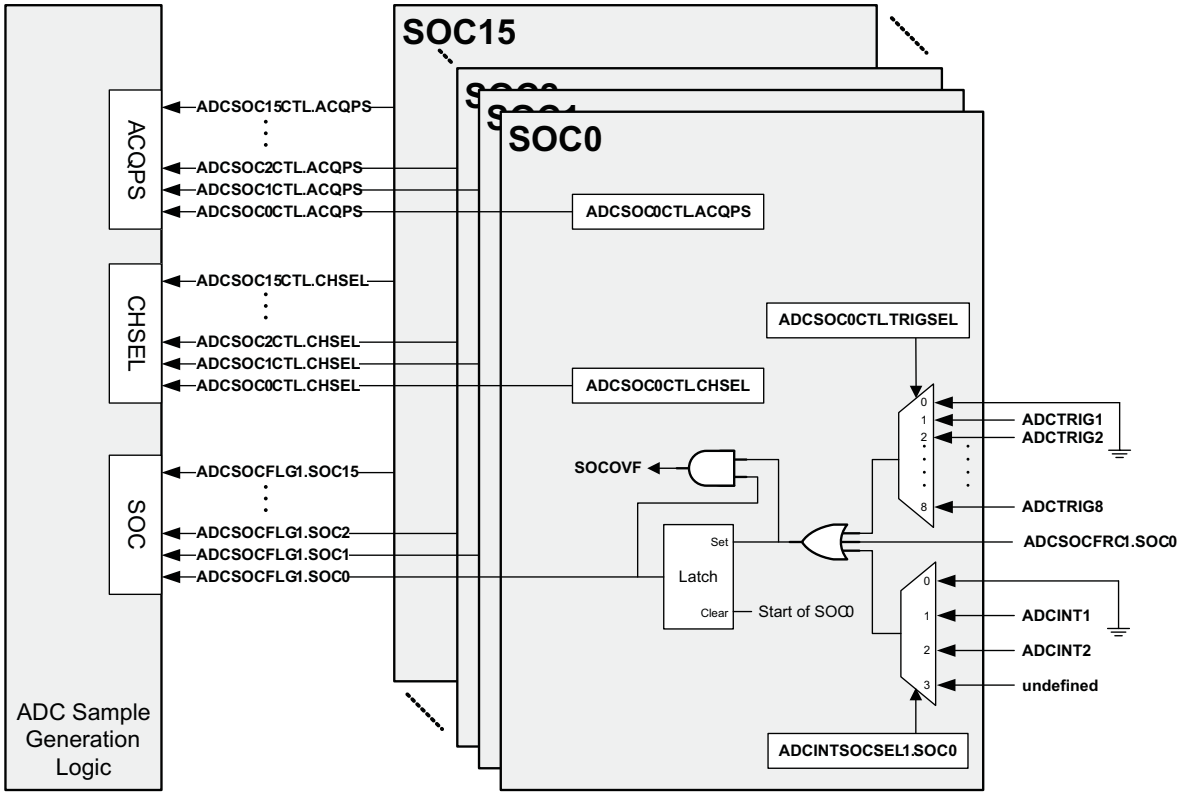
10.3.3 SOC Principle of Operation

Contrary to previous ADC types, this ADC is not sequencer based. Instead, it is SOC based. The term SOC is the configuration set defining the single conversion of a single channel. In that set there are three configurations: the trigger source that starts the conversion, the channel to convert, and the acquisition (sample) window size. Each SOC is independently configured and can have any combination of the trigger, channel, and sample window size available. Multiple SOCs can be configured for the same trigger, channel, and/or acquisition window as desired. This provides a very flexible means of configuring conversions ranging from individual samples of different channels with different triggers, to oversampling the same channel using a single trigger, to creating your own series of conversions of different channels all from a single trigger.

The trigger source for SOCx is configured by a combination of the TRIGSEL field in the ADCSOCxCTL register, the appropriate bits in the ADCINTSOCSEL1 or ADCINTSOCSEL2 register, and setting the correct bits in the TRIGxSEL registers. Software can also force an SOC event with the ADCSOCFRC1 register. The channel and sample window size for SOCx are configured with the CHSEL and ACQPS fields of the ADCSOCxCTL register.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-11. SOC Block Diagram



For example, to configure a single conversion on channel ADCINA1 to occur when the ePWM3 timer reaches its period match you must first setup ePWM3 to output an SOCA or SOCB signal on a period match. See the *Enhanced Pulse Width Modulator Module (EPWM)* chapter on how to do this. For this example, we'll use SOCA. Then, setup one of the SOCs using its ADCSOCxCTL register. It makes no difference which SOC we choose, so we'll use SOC0. It also makes no difference which ADC trigger we choose, so we'll use ADC trigger 1. The fastest allowable sample window for the ADC is 7 cycles. Choosing the fastest time for the sample window and channel ADCINA1 for the channel to convert, we'll set the ACQPS field to 6 and the CHSEL field to 1. To use ePWM3 for the SOC0 trigger, we'll set the TRIGSEL field to 5 to choose ADC trigger 1 and the TRIG1SEL register to 11 to choose ePWM3 SOCA as the trigger source. TRIGxSEL registers are located in the Analog System Control register set. The resulting values written into the registers will be:

```
ADCSOC0CTL = 2846h;          // (ACQPS=6, CHSEL=1, TRIGSEL=5)
TRIG1SEL = 000Bh             // (TRIG1SEL=11)
```

When configured as such, a single conversion of ADCINA1 will be started on an ePWM3 SOCA event with the resulting value stored in the ADCRESULT0 register.

If instead ADCINA1 needed to be oversampled by 3X, then SOC1, SOC2, and SOC3 could all be given the same configuration as SOC0.

```
ADCSOC1CTL = 2846h;          // (ACQPS=6, CHSEL=1, TRIGSEL=5)
ADCSOC2CTL = 2846h;          // (ACQPS=6, CHSEL=1, TRIGSEL=5)
ADCSOC3CTL = 2846h;          // (ACQPS=6, CHSEL=1, TRIGSEL=5)
TRIG1SEL = 000Bh             // (TRIG1SEL=11)
```


Copyright © 2012–2019, Texas Instruments Incorporated

When configured as such, four conversions of ADCINA1 will be started in series on an ePWM3 SOCA event with the resulting values stored in the ADCRESULT0 – ADCRESULT3 registers.

Another application may require 3 different signals to be sampled from the same trigger. This can be done by simply changing the CHSEL field for SOC0-SOC2 while leaving the TRIGSEL field and TRIG1SEL register unchanged.

```
ADCSOC0CTL = 2846h;   // (ACQPS=6, CHSEL=1, TRIGSEL=5)
ADCSOC1CTL = 2886h;   // (ACQPS=6, CHSEL=2, TRIGSEL=5)
ADCSOC2CTL = 28C6h;   // (ACQPS=6, CHSEL=3, TRIGSEL=5)
TRIG1SEL = 000Bh      // (TRIG1SEL=11)
```

When configured this way, three conversions will be started in series on an ePWM3 SOCA event. The result of the conversion on channel ADCINA1 will show up in ADCRESULT0. The result of the conversion on channel ADCINA2 will show up in ADCRESULT1. The result of the conversion on channel ADCINA3 will show up in ADCRESULT2. The channel converted and the trigger have no bearing on where the result of the conversion shows up. The RESULT register is associated with the SOC.

NOTE: These examples are incomplete. Clocks must be enabled and the ADC must be powered to work correctly. See Section 10.5 for more details on how to configure Analog Subsystem clocks. For the power up sequence of the ADC, see Section 10.3.8.

Table 10-2 shows the configuration options for the TRIGxSEL registers.

Table 10-2. TRIGxSEL Trigger Options

TRIGxSEL Bits	Trigger Source	Peripheral
00000	No Trigger Enabled (Default)	
00001	TINT0	CPU Timer 0
00010	TINT1	CPU Timer 1
00011	TINT2	CPU Timer 2
00100	ADCEXTTRIG	C28 GPIO MUX
00101	EPWM1SOCA	EPWM1
00110	EPWM1SOCB	
00111	EPWM1SYNC	
01000	EPWM2SOCA	EPWM2
01001	EPWM2SOCB	
01010	EPWM2SYNC	
01011	EPWM3SOCA	EPWM3
01100	EPWM3SOCB	
01101	EPWM3SYNC	
01110	EPWM4SOCA	EPWM4
01111	EPWM4SOCB	
10000	EPWM4SYNC	
10001	EPWM5SOCA	EPWM5
10010	EPWM5SOCB	
10011	EPWM5SYNC	
10100	EPWM6SOCA	EPWM6
10101	EPWM6SOCB	
10110	EPWM6SYNC	
10111	EPWM7SOCA	EPWM7
11000	EPWM7SOCB	
11001	EPWM7SYNC	

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-2. TRIGxSEL Trigger Options (continued)

TRIGxSEL Bits	Trigger Source	Peripheral
11010	EPWM8SOCA	EPWM8
11011	EPWM8SOCB	
11100	EPWM8SYNC	
11101	EPWM9SOCA	EPWM9
11110	EPWM9SOCB	
11111	EPWM9SYNC	

10.3.3.1 ADC Acquisition (Sample and Hold) Window

External drivers vary in their ability to drive an analog signal quickly and effectively. Some circuits require longer times to properly transfer the charge into the sampling capacitor of an ADC. To address this, the ADC supports control over the sample window length for each individual SOC configuration. Each ADCSOCxCTL register has a 6-bit field, ACQPS, that determines the sample and hold (S/H) window size. The value written to this field is one less than the number of cycles desired for the sampling window for that SOC. Thus, a value of 15 in this field will give 16 clock cycles of sample time. The minimum number of sample cycles allowed is 7 (ACQPS=6). The total sampling time is found by adding the sample window size to the conversion time of the ADC, 13 ADC clocks. Examples of various sample times are shown below in Table 10-3.

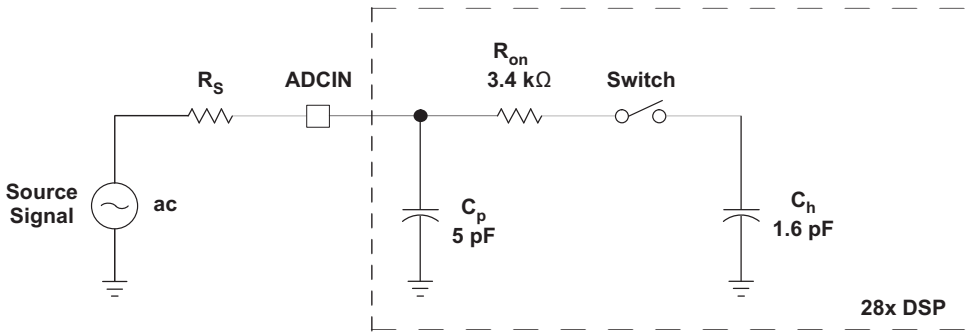
Table 10-3. Sample timings with different values of ACQPS

ADC Clock	ACQPS	Sample Window	Conversion Time (13 cycles)	Total Time to Process Analog Voltage[1]
37.5MHz	6	187ns	347ns	534ns
37.5MHz	25	693ns	347ns	1040ns
30MHz	6	233ns	433ns	666ns
30MHz	25	867ns	433ns	1300ns
25MHz	6	280ns	520ns	800ns
25MHz	25	1040ns	520ns	1560ns

(1) The total times are for a single conversion and do not include pipelining effects that increase the average speed over time.

As shown in Figure 10-12 , the ADCIN pins can be modeled as an RC circuit. With VREFLO connected to ground, a voltage swing from 0 to 3.3v on ADCIN yields a typical RC time constant of 2ns.

Figure 10-12. ADCINx Input Model



Typical Values of the Input Circuit Components:

Switch Resistance (R_{on}): 3.4 kΩ
Sampling Capacitor (C_h): 1.6 pF
Parasitic Capacitance (C_p): 5 pF
Source Resistance (R_S): 50 Ω

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.3.2 Trigger Operation

Each SOC can be configured to start on one of many input triggers. Multiple SOC's can be configured for the same channel if desired. Following is a list of the available input triggers:

- Software
- CPU Timers 0/1/2 interrupts
- External GPIO through GPTRIP2SEL register
- ePWM1-9 SOCA and SOCB

See the ADCSOCxCTL Register Bit Definitions and the TRIGxSEL Register Bit Definitions for the configuration details of these triggers.

Additionally ADCINT1 and ADCINT2 can be fed back to trigger another conversion. This configuration is controlled in the ADCINTSOCSEL1/2 registers. This mode is useful if a continuous stream of conversions is desired. See Section 10.3.7 for information on the ADC interrupt signals.

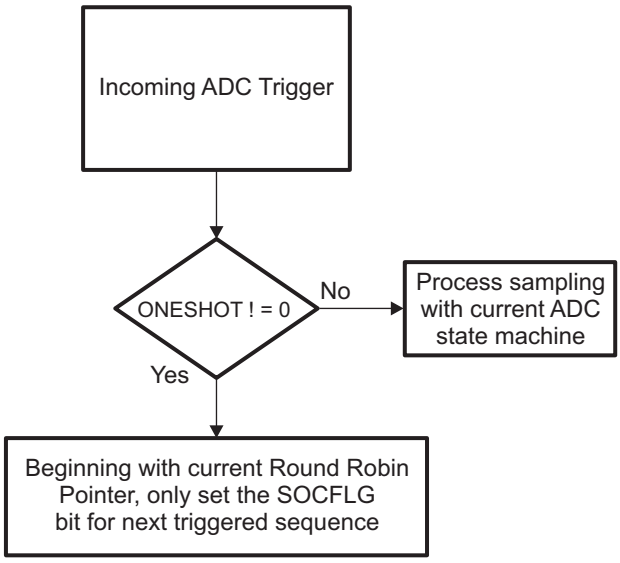
10.3.3.3 Channel Selection

Each SOC can be configured to convert any of the available ADCIN input channels. When an SOC is configured for sequential sampling mode, the four bit CHSEL field of the ADCSOCxCTL register defines which channel to convert. When an SOC is configured for simultaneous sampling mode, the most significant bit of the CHSEL field is dropped and the lower three bits determine which pair of channels are converted.

10.3.4 ONESHOT Single Conversion Support

This mode will allow you to perform a single conversion on the next triggered SOC in the round robin scheme. The ONESHOT mode is only valid for channels present in the round robin wheel. Channels which are not configured for triggered SOC in the round robin scheme will get priority based on contents of the SOCPRIORITY field in the ADCSOCPRIORITYCTL register.

Figure 10-13. ONESHOT Single Conversion



The effect of ONESHOT mode on Sequential Mode and Simultaneous Mode is explained below.

Sequential mode: Only the next active SOC in RR mode (one up from current RR pointer) will be allowed to generate SOC; all other triggers for other SOC slots will be ignored.

Simultaneous mode: If current RR pointer has SOC with simultaneous enabled; active SOC will be incremented by 2 from the current RR pointer. This is because simultaneous mode will create result for SOCx and SOCx+1, and SOCx+1 will never be triggered by the user.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: ONESHOT = 1 and SOCPRIORITY = 10h is not a valid combination for above implementation reasons. This should not be a desired mode of operation by the user in any case. The limitation of the above is that the next SOCs must eventually be triggered, or else the ADC will not generate new SOCs for other out-of-order triggers. Any non-orthogonal channels should be placed in the priority mode which is unaffected by ONESHOT mode

10.3.5 ADC Conversion Priority

When multiple SOC flags are set at the same time, one of two forms of priority determines the order in which they are converted. The default priority method is round robin. In this scheme, no SOC has an inherent higher priority than another. Priority depends on the round robin pointer (RRPOINTER). The RRPOINTER reflected in the ADCSOCPRIORITYCTL register points to the last SOC converted. The highest priority SOC is given to the next value greater than the RRPOINTER value, wrapping around back to SOC0 after SOC15. At reset the value is 32 since 0 indicates a conversion has already occurred. When RRPOINTER equals 32 the highest priority is given to SOC0. The RRPOINTER is reset by a device reset, when the ADCCTL1.RESET bit is set, or when the SOCPRICTL register is written.

An example of the round robin priority method is given in Figure 10-14.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-14. Round Robin Priority Example

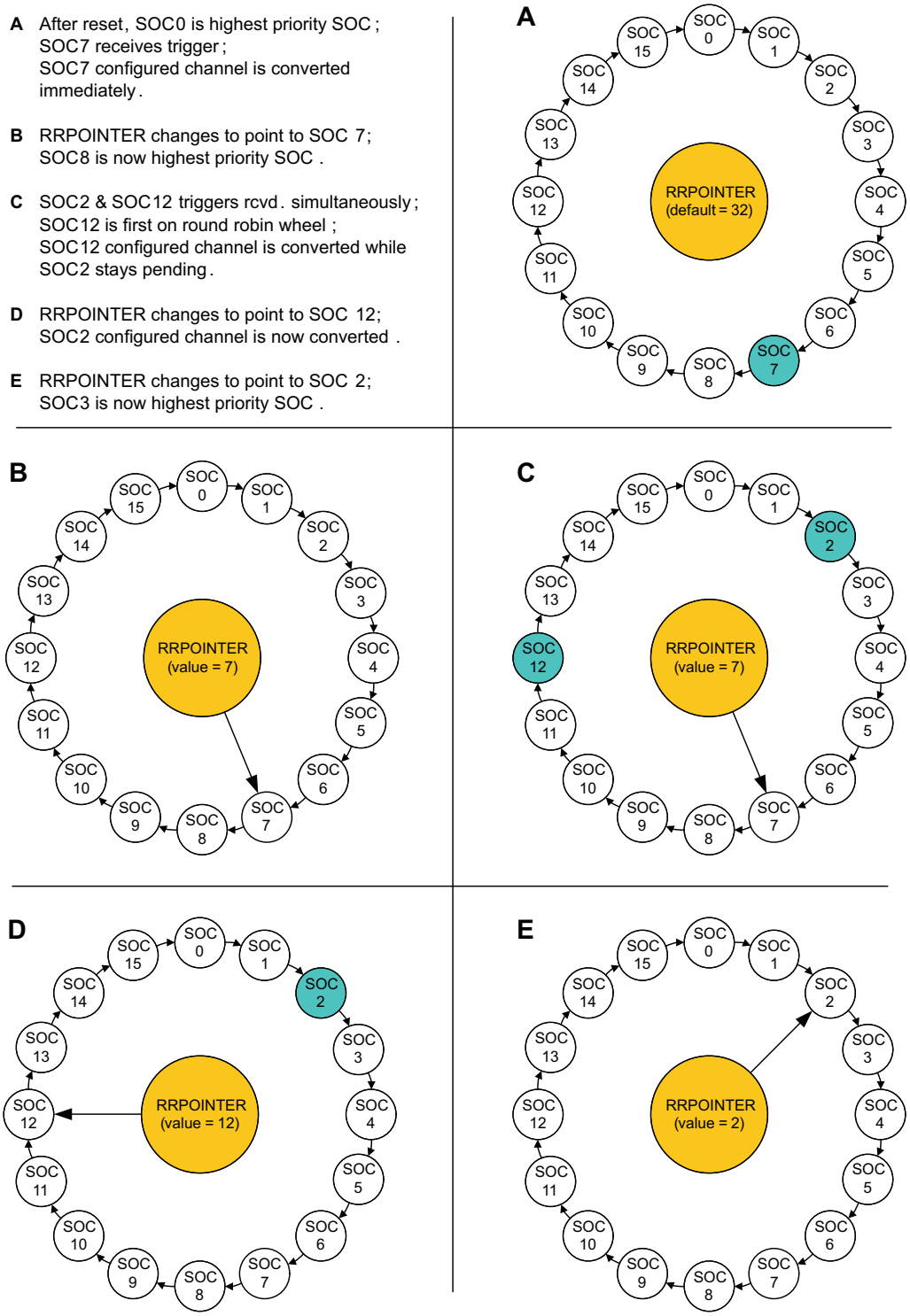
A After reset, SOC0 is highest priority SOC; SOC7 receives trigger; SOC7 configured channel is converted immediately.

B RRPOINTER changes to point to SOC 7; SOC8 is now highest priority SOC .

C SOC2 & SOC12 triggers rcvd. simultaneously; SOC12 is first on round robin wheel ; SOC12 configured channel is converted while SOC2 stays pending.

D RRPOINTER changes to point to SOC 12; SOC2 configured channel is now converted .

E RRPOINTER changes to point to SOC 2; SOC3 is now highest priority SOC .



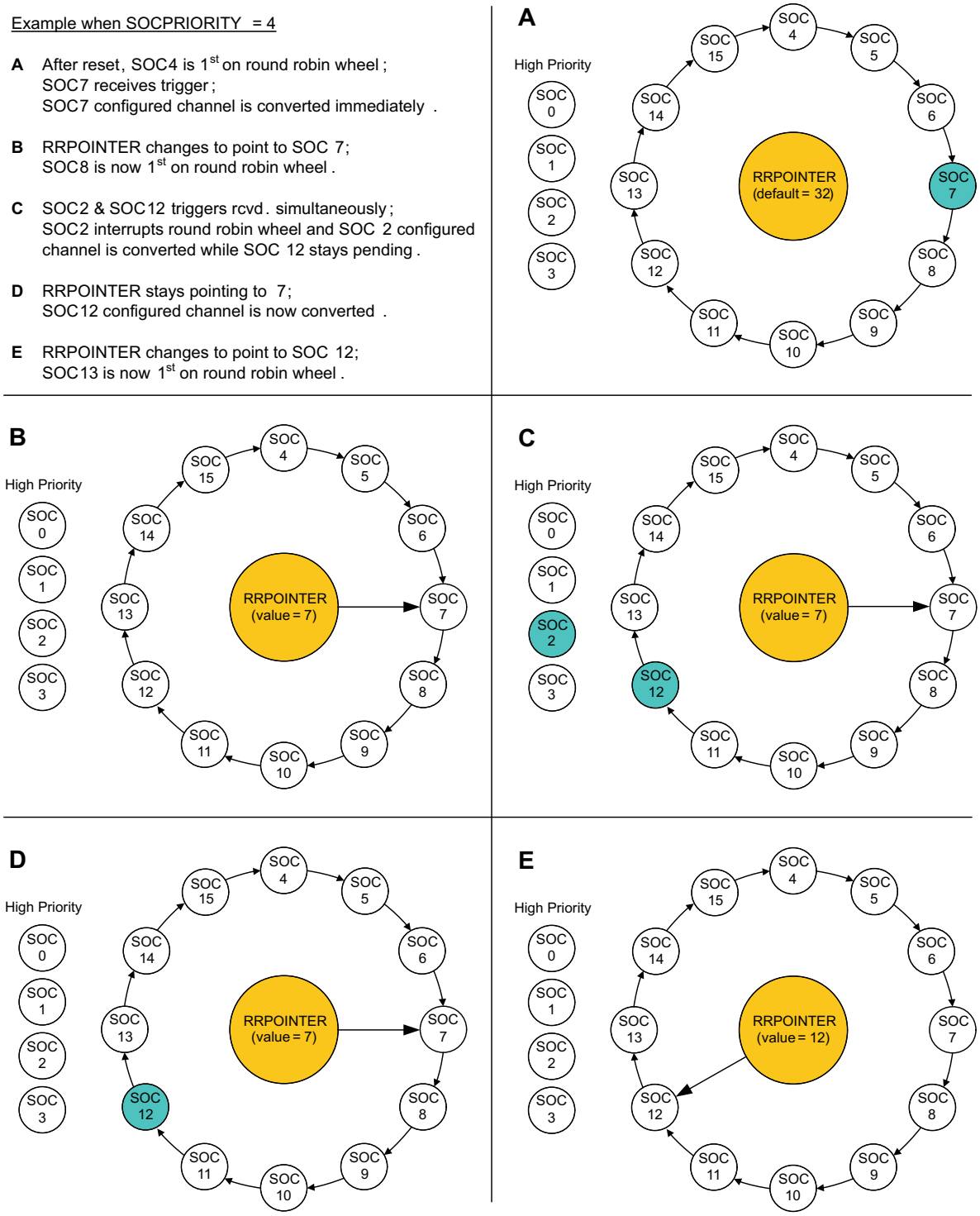
The SOCPRIORITY field in the ADCSOCPRIORITYCTL register can be used to assign high priority from a single to all of the SOC's. When configured as high priority, an SOC will interrupt the round robin wheel after any current conversion completes and insert itself in as the next conversion. After its conversion completes, the round robin wheel will continue where it was interrupted. If two high priority SOC's are triggered at the same time, the SOC with the lower number will take precedence.

Copyright © 2012–2019, Texas Instruments Incorporated

High priority mode is assigned first to SOC0, then in increasing numerical order. The value written in the SOCPRIORITY field defines the first SOC that is not high priority. In other words, if a value of 4 is written into SOCPRIORITY, then SOC0, SOC1, SOC2, and SOC3 are defined as high priority, with SOC0 the highest.

An example using high priority SOC's is given in Figure 10-15 .

Figure 10-15. High Priority Example



Copyright © 2012–2019, Texas Instruments Incorporated

10.3.6 Simultaneous Sampling Mode

In some applications it is important to keep the delay between the sampling of two signals minimal. The ADC contains dual sample and hold circuits to allow two different channels to be sampled simultaneously. Simultaneous sampling mode is configured for a pair of SOCx's with the ADCSAMPLEMODE register. The even numbered SOCx and the following odd numbered SOCx (SOC0 and SOC1) are coupled together with one enable bit (SIMULEN0, in this case). The coupling behavior is as follows:

- Either SOCx's trigger will start a pair of conversions.
- The pair of channels converted will consist of the A-channel and the B-channel corresponding to the value of the CHSEL field of the triggered SOCx. The valid values in this mode are 0-7.
- Both channels will be sampled simultaneously.
- The A channel will always convert first.
- The even EOCx pulse will be generated based off of the A-channel conversion, the odd EOCx pulse will be generated off of the B-channel conversion. See Section 10.3.7 for an explanation of the EOCx signals.
- The result of the A-channel conversion is placed in the even ADCRESULTx register and the result of the B-channel conversion is written to the odd ADCRESULTx register.

NOTE: If EOCx pulses are configured to generate interrupts, and ADC triggers are set up for both the even and odd SOCs in the same pair, two interrupts will be generated. If this is not the desired behavior, only configure the trigger for either the A channel or B channel in the pair to receive one interrupt.

For example, if the ADCSAMPLEMODE.SIMULEN0 bit is set, and SOC0 is configured as follows:

CHSEL = 2 (ADCINA2/ADCINB2 pair)

TRIGSEL = 5 (ADCTRIG1)

TRIG1SEL = 5 (ePWM1.ADCSOCA)

When the ePWM1 sends out an ADCSOCA trigger, both ADCINA2 and ADCINB2 will be sampled simultaneously (assuming priority). Immediately after, the ADCINA2 channel will be converted and its value will be stored in the ADCRESULT0 register. Depending on the ADCCTL1.INTPULSEPOS setting, the EOC0 pulse will either occur when the conversion of ADCINA2 begins or completes. Then the ADCINB2 channel will be converted and its value will be stored in the ADCRESULT1 register. Depending on the ADCCTL1.INTPULSEPOS setting, the EOC1 pulse will either occur when the conversion of ADCINB2 begins or completes.

Typically in an application it is expected that only the even SOCx of the pair will be used. However, it is possible to use the odd SOCx instead, or even both. In the latter case, both SOCx triggers will start a conversion. Therefore, caution is urged as both SOCx's will store their results to the same ADCRESULTx registers, possibly overwriting each other.

The rules of priority for the SOCx's remain the same as in sequential sampling mode.

Section 10.3.13 shows the timing of simultaneous sampling mode.

10.3.7 EOC and Interrupt Operation

Just as there are 16 independent SOCx configuration sets, there are 16 EOCx pulses. In sequential sampling mode, the EOCx is associated directly with the SOCx. In simultaneous sampling mode, the even and the following odd EOCx pair are associated with the even and the following odd SOCx pair, as described in Section 10.3.6. Depending on the ADCCTL1.INTPULSEPOS setting, the EOCx pulse will occur either at the beginning of a conversion or the end. See Section 10.3.13 for exact timings on the EOCx pulses.

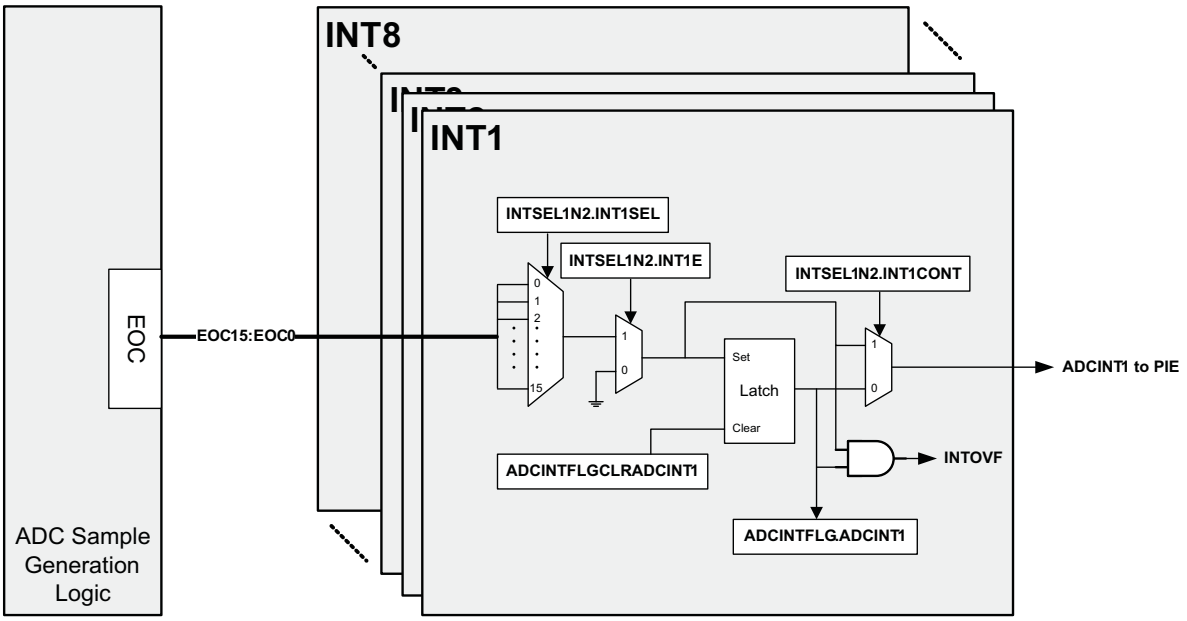
The ADC contains eight interrupts that can be flagged and/or passed on to the PIE and NVIC. Each of these interrupts can be configured to accept any of the available EOCx signals as its source. The configuration of which EOCx is the source is done in the INTSELxNy registers. Additionally, the ADCINT1 and ADCINT2 signals can be configured to generate an SOCx trigger. This is beneficial to creating a continuous stream of conversions.

There are a total of 8 interrupts available for both ADC1 and ADC2. Each ADC does not have its own set of 8 interrupts. These resources must be shared.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-16 shows a block diagram of the interrupt structure of the ADC.

Figure 10-16. Interrupt Structure



10.3.8 Power Up Sequence

The ADC resets to the ADC off state. Before writing to any of the ADC registers the Analog Subsystem clocks must be enabled. For a description on how to enable these clocks, see Section 10.5. When powering up the ADC, use the following sequence:

1. If an external reference is desired, enable this mode using bit 3 (ADCREFSEL) in the ADCCTL1 register.
2. Power up the reference, bandgap, and analog circuits together by setting bits 7-5 (ADCPWDN, ADCBGPWD, ADCREFPWD) in the ADCCTL1 register. Intermediary states are not currently supported.
3. Enable the ADC by setting bit 14 (ADCENABLE) of the ADCCTL1 register.
4. Before performing the first conversion, a delay of 1 millisecond after step 2 is required.

Alternatively, steps 1 through 3 can be performed simultaneously.

When powering down the ADC, all three bits in step 2 can be cleared simultaneously. The ADC power levels must be controlled via software and they are independent of the state of the device power modes.

NOTE: This type ADC requires a 1ms delay after all of the circuits are powered up. This differs from the previous type ADC's.

10.3.9 ADC Calibration

Inherent in any converter is a zero offset error and a full scale gain error. The ADC is factory calibrated at 25-degrees Celsius to correct both of these while allowing the user to modify the offset correction for any application environmental effects, such as the ambient temperature. Except under certain emulation conditions, or unless a modification from the factory settings is desired, the user is not required to perform any specific action. The ADC will be properly calibrated during the device boot process.

NOTE: If the system is reset or the ADC module is reset using Bit 15 (RESET) from the ADC Control Register 1, the Device_cal() routine must be repeated.

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.9.1 Factory Settings and Calibration Function

During the fabrication and test process Texas Instruments calibrates several ADC settings along with a couple of internal oscillator settings. These settings are embedded into the TI reserved OTP memory as part of a C-callable function named Device_cal(). This function writes the factory settings into their respective active registers. Until this occurs, the ADC and the internal oscillators will not adhere to their specified parameters. The user must ensure the trim settings are written to their respective registers to ensure the ADC and the internal oscillators meet the specifications in the datasheet. This can be done either by calling this function manually or in the application itself, or by a direct write via CCS. A gel function is provided as part of the controlSUITE™ to accomplish this.

Texas Instruments cannot guarantee the parameters specified in the datasheet if a value other than the factory settings contained in the TI reserved OTP memory is written into the ADC trim registers.

10.3.9.2 ADC Zero Offset Calibration

Zero offset error is defined as the resultant digital value that occurs when converting a voltage at VREFLO. This base error affects all conversions of the ADC and together with the full scale gain and linearity specifications, determine the DC accuracy of a converter. The zero offset error can be positive, meaning that a positive digital value is output when VREFLO is presented, or negative, meaning that a voltage higher than a one step above VREFLO still reads as a digital zero value. To correct this error, the two's complement of the error is written into the ADCOFFTRIM register. The value contained in this register will be applied before the results are available in the ADC result registers. This operation is fully contained within the ADC core, so the timing for the results will not be affected and the full dynamic range of the ADC will be maintained for any trim value. Calling the Device_cal() function writes the ADCOFFTRIM register with the factory calibrated offset error correction, but the user can modify the ADCOFFTRIM register to compensate for additional offset error induced by the application environment. This can be done without sacrificing an ADC channel by using the VREFLOCONV bit in the ADCCTRL1 register.

Use the following procedure to re-calibrate the ADC offset:

1. **Set ADCOFFTRIM to 80 (50h)**. This adds an artificial offset to account for negative offset that may reside in the ADC core.
2. **Set ADCCTL1.VREFLOCONV to 1.** This internally connects VREFLO to input channel B5. See the ADCCTL1 register description for more details.
3. **Perform multiple conversions on B5 (sample VREFLO) and take an average to account for board noise.** See Section 10.3.3 on how to setup and initiate the ADC to sample B5.
4. **Set ADCOFFTRIM to 80 (50h) minus the average obtained in step 3.** This removes the artificial offset from step 1 and creates a two's compliment of the offset error.
5. **Set ADCCTL1.VREFLOCONV to 0.** This connects B5 back to the external ADCINB5 input pin.

NOTE: The "AdcOffsetSelfCal()" function located in F28M35x_Adc.c in the common header files performs these steps.

10.3.9.3 ADC Full Scale Gain Calibration

Gain error occurs as an incremental error as the voltage input is increased. Full scale gain error occurs at the maximum input voltage. As in offset error, gain error can be positive or negative. A positive full scale gain error means that the full scale digital result is reached before the maximum voltage is input. A negative full scale error implies that the full digital result will never be achieved. The calibration function Device_cal() writes a factory trim value to correct the ADC full scale gain error into the ADCREFTRIM register. This register should not be modified after the Device_cal() function is called.

10.3.9.4 ADC Bias Current Calibration

To further increase the accuracy of the ADC, the calibration function Device_cal() also writes a factory trim value to an ADC register for the ADC bias currents. This register should not be modified after the Device_cal() function is called.

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.10 Internal/External Reference Voltage Selection

10.3.10.1 Internal Reference Voltage

The ADC can operate in two different reference modes, selected by the ADCCTL1.ADCREFSEL bit. By default the internal bandgap is chosen to generate the reference voltage for the ADC. This will convert the voltage presented according to a fixed scale 0 to 3.3v range. The equation governing conversions in this mode is:

Digital Value = 0	when Input ≤ 0v
Digital Value = 4096 [(Input – VREFLO)/3.3v]	when 0v < Input < 3.3v
Digital Value = 4095,	when Input ≥ 3.3v

*All fractional values are truncated

**VREFLO must be tied to ground in this mode. This is done internally on some devices.

10.3.10.2 External Reference Voltage

To convert the voltage presented as a ratiometric signal, the external VREFHI/VREFLO pins should be chosen to generate the reference voltage. In contrast with the fixed 0 to 3.3v input range of the internal bandgap mode, the ratiometric mode has an input range from VREFLO to VREFHI. Converted values will scale to this range. For instance, if VREFLO is set to 0.5v and VREFHI is 3.0v, a voltage of 1.75v will be converted to the digital result of 2048. See the device datasheet for the allowable ranges of VREFLO and VREFHI. On some devices VREFLO is tied to ground internally, and hence limited to 0v. The equation governing the conversions in this mode is:

Digital Value = 0	when Input ≤ VREFLO
Digital Value = 4096 [(Input – VREFLO)/(VREFHI – VREFLO)]	when VREFLO < Input < VREFHI
Digital Value = 4095,	when Input ≥ VREFHI

*All fractional values are truncated

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.11 ADC Registers

This section contains the ADC registers and bit definitions with the registers grouped by function. All of the ADC registers are located in Peripheral Frame 2 except the ADCRESULTx registers, which are found in Peripheral Frame 0. See the device datasheet for specific addresses.

Table 10-4. ADC Registers[1]

Register Name	Address Offset	Size (x16)	Description
ADC1	7100h - 7174h	1	Analog-to-Digital Converter 1
ADC2	7180h - 71FFh	1	Analog-to-Digital Converter 2
ADC1 Results	0B00h - 0B0Fh	1	Analog-to-Digital Converter 1 Results
ADC2 Results	0B40h - 0B4Fh	1	Analog-to-Digital Converter 2 Results
ADC1 Results M3	5000 1600h - 5000 161Fh	1	Analog-to-Digital Converter 1 Results (M3)
ADC2 Results M3	5000 1680h - 5000 168Fh	1	Analog-to-Digital Converter 2 Results (M3)

(1) The second set of ADC result registers are mapped to the M3 memory space. This gives both cores access to the ADC conversion results.

Table 10-5. ADC Configuration and Control Registers (AdcRegs and AdcResult):

Register Name	Address Offset	Size (x16)	Description
ADCCTL1	00h	1	Control 1 Register[1]
ADCCTL2	01h	1	Control 2 registerSection 10.3.11.2
ADCINTFLG	04h	1	Interrupt Flag Register
ADCINTFLGCLR	05h	1	Interrupt Flag Clear Register
ADCINTOVF	06h	1	Interrupt Overflow Register
ADCINTOVFCLR	07h	1	Interrupt Overflow Clear Register
INTSEL1N2	08h	1	Interrupt 1 and 2 Selection Register[1]
INTSEL3N4	09h	1	Interrupt 3 and 4 Selection Register[1]
INTSEL5N6	0Ah	1	Interrupt 5 and 6 Selection Register[1]
INTSEL7N8	0Bh	1	Interrupt 7 and 8 Selection Register[1]
SOCPRICTL	10h	1	SOC Priority Control Register[1]
ADCSAMPLEMODE	12h	1	Sampling Mode Register[1]
ADCINTSOCSEL1	14h	1	Interrupt SOC Selection 1 Register (for 8 channels)[1]
ADCINTSOCSEL2	15h	1	Interrupt SOC Selection 2 Register (for 8 channels)[1]
ADCSOCFLG1	18h	1	SOC Flag 1 Register (for 16 channels)
ADCSOCFRC1	1Ah	1	SOC Force 1 Register (for 16 channels)
ADCSOCOVF1	1Ch	1	SOC Overflow 1 Register (for 16 channels)
ADCSOCOVFCLR1	1Eh	1	SOC Overflow Clear 1 Register (for 16 channels)
ADCSOC0CTL - ADCSOC15CTL	20h - 2Fh	1	SOC0 Control Register to SOC15 Control Register[1]
ADCREFTRIM	40h	1	Reference Trim Register[1]
ADCOFFTRIM	41h	1	Offset Trim Register[1]
ADCREV – reserved	4Fh	1	Revision Register
ADCRESULT0 - ADCRESULT15	00h - 0Fh[2]	1	ADC Result 0 Register to ADC Result 15 Register

(1) This register is EALLOW protected.
(2) The base address of the ADCRESULT registers differs from the base address of the other ADC registers. In the header files, the ADCRESULT registers are found in the AdcResult register file, not AdcRegs.

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.11.1 ADC Control Register 1 (ADCCTL1)

NOTE: The following ADC Control Register is EALLOW protected.

Figure 10-17. ADC Control Register 1 (ADCCTL1) (Address Offset 00h)

15	14	13	12				8
RESET	ADCENABLE	ADCBSY	ADCBSYCHN				
R-0/W-1	R-1	R-0	R-0				
7	6	5	4	3	2	1	0
ADCPWDN	ADCBGPWD	ADCREFPWD	Reserved	ADCREFSEL	INTPULSEPOS	VREFLO CONV	Reserved
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; R-0/W-1 = always read as 0, write 1 to set; -*n* = value after reset

Table 10-6. ADC Control Register 1 (ADCCTL1) Field Descriptions

Bit	Field	Value	Description
15	RESET		ADC module software reset. This bit causes a master reset on the entire ADC module. All register bits and state machines are reset to the initial state as occurs when the device reset pin is pulled low (or after a power-on reset). This is a one-time-effect bit, meaning this bit is self-cleared immediately after it is set to 1. Read of this bit always returns a 0.
		0	No effect
		1	Resets the entire ADC module (bit is then set back to 0 by ADC logic)
			Note: The ADC module is reset during a system reset. If an ADC module reset is desired at any other time, you can do so by writing a 1 to this bit. **Note:** If the system is reset or the ADC module is reset using Bit 15 (RESET) from the ADC Control Register 1, the Device_cal() routine must be repeated.
14	ADCENABLE		ADC Enable
		0	ADC disabled (**does not** power down ADC)
		1	ADC Enabled. Must set before an ADC conversion (recommend that it be set directly after setting ADC power-up bits
13	ADCBSY		ADC Busy
			Set when ADC SOC is generated, cleared per below. Used by the ADC state machine to determine if ADC is avaliable to sample.
			Sequential Mode: Cleared 4 ADC clocks after negative edge of S/H pulse
			Simultaneous Mode: Cleared 14 ADC clocks after negative edge of S/H pulse
		0	ADC is available to sample next channel
		1	ADC is busy and cannot sample another channel

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-6. ADC Control Register 1 (ADCCTL1) Field Descriptions (continued)

Bit	Field	Value	Description
12-8	ADCBSYCHN		When ADCBSY = 0: holds the value of the last executed ADC SOC
			When ADCBSY = 1: reflects the ADC SOC currently being processed
		00h	ADC SOC0 is currently processing or was last ADC SOC executed
		01h	ADC SOC1 is currently processing or was last ADC SOC executed
		02h	ADC SOC2 is currently processing or was last ADC SOC executed
		03h	ADC SOC3 is currently processing or was last ADC SOC executed
		04h	ADC SOC4 is currently processing or was last ADC SOC executed
		05h	ADC SOC5 is currently processing or was last ADC SOC executed
		06h	ADC SOC6 is currently processing or was last ADC SOC executed
		07h	ADC SOC7 is currently processing or was last ADC SOC executed
		08h	ADC SOC8 is currently processing or was last ADC SOC executed
		09h	ADC SOC9 is currently processing or was last ADC SOC executed
		0Ah	ADC SOC10 is currently processing or was last ADC SOC executed
		0Bh	ADC SOC11 is currently processing or was last ADC SOC executed
		0Ch	ADC SOC12 is currently processing or was last ADC SOC executed
		0Dh	ADC SOC13 is currently processing or was last ADC SOC executed
		0Eh	ADC SOC14 is currently processing or was last ADC SOC executed
		0Fh	ADC SOC15 is currently processing or was last ADC SOC executed
		1xh	Invalid value
7	ADCPWDN		ADC power down (active low).
			This bit controls the power up and power down of all the analog circuitry inside the analog core except the bandgap and reference circuitry
		0	All analog circuitry inside the core except the bandgap and referencce circuitry is powered down
		1	The analog circuitry inside the core is powered up
6	ADCBGPWD		Bandgap circuit power down (active low)
		0	Bandgap circuitry is powered down
		1	Bandgap buffer's curcuitry inside core is powered up
5	ADCREFPWD		Reference buffers circuit power down (active low)
		0	Reference buffers circuitry is powered down
		1	Reference buffers circuitry inside the core is powered up
4	Reserved	0	Reserved
3	ADCREFSEL		Internal/external reference select
		0	Internal Bandgap used for reference generation
		1	External VREFHI/VREFLO pins used for reference generation.
2	INTPULSEPOS		INT Pulse Generation control
		0	INT pulse generation occurs when ADC begins conversion (neg edge of sample pulse od the sampled signal)
		1	INT pulse generation occurs 1 cycle prior to ADC result latching into its result register
1	VREFLOCONV		VREFLO Convert.
			When enabled, internally connects VREFLO to the ADC channel B5 and disconnects the ADCINB5 pin from the ADC. Whether the pin ADCINB5 exists on the device does not affect this function. Any external circuitry on the ADCINB5 pin is unaffected by this mode.
		0	ADCINB5 is passed to the ADC module as normal, VREFLO connection to ADCINB5 is disabled
		1	VREFLO internally connected to the ADC for sampling
0	Reserved		Reserved

10.3.11.2 ADC Control Register 2 (ADCCTL2)

The register and description are shown below.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-18. ADC Control Register 2 (ADCCTL2) (Address Offset 01h)

15	9	8
Reserved		
R-0		

7	2	1	0
Reserved		ADCNONOVERLAP	CLKDIV2EN
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-7. ADC Control Register 2 (ADCCTL2) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reserved
1	ADCNONOVERLAP		ADCNONOVERLAP Control bit.
		0	Overlap of sample and conversion is allowed.
		1	Overlap of sample is not allowed.
0	CLKDIV2EN		When enabled, divides the ADC input clock by 2. When running /2 ADCCLK, scale the minimum sample duration accordingly to meet 187ns for better throughput.
		0	ADC clock = ACIB clock
		1	ADC clock = ACIB clock/2

10.3.11.3 ADC Interrupt Registers

The registers and descriptions are shown below.

Figure 10-19. ADC Interrupt Flag Register (ADCINTFLG) (Address Offset 04h)

15	9	8
Reserved		
R-0		

7	6	5	4	3	2	1	0
ADCINT8	ADCINT7	ADCINT6	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-8. ADC Interrupt Flag Register (ADCINTFLG) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7-0	ADCINTx (x = 8 to 1)		ADC Interrupt Flag Bits: Reading this bit indicates if an ADCINT pulse was generated
		0	No ADC interrupt pulse generated
		1	ADC Interrupt pulse generated
			If the ADC interrupt is placed in continuous mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINTFLGCLR register. The ADCINTOVF flag will be set if EOC events are generated while the ADCINTFLG flag is set. Both ADCINTFLG and ADCINTOVF flags must be cleared before normal interrupt operation can resume.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-20. ADC Interrupt Flag Clear Register (ADCINTFLGCLR) (Address Offset 05h)

15						9	8
Reserved							
R-0							
7	6	5	4	3	2	1	0
ADCINT8	ADCINT7	ADCINT6	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-9. ADC Interrupt Flag Clear Register (ADCINTFLGCLR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7-0	ADCINTx (x = 8 to 1)		ADC interrupt Flag Clear Bit
		0	No action.
		1	Clears respective flag bit in the ADCINTFLG register. If software tries to set this bit on the same clock cycle that hardware tries to set the flag bit in the ADCINTFLG register, then hardware has priority and the ADCINTFLG bit will be set. In this case the overflow bit in the ADCINTOVF register will not be affected regardless of whether the ADCINTFLG bit was previously set or not.

Figure 10-21. ADC Interrupt Overflow Register (ADCINTOVF) (Address Offset 06h)

15						9	8
Reserved							
R-0							
7	6	5	4	3	2	1	0
ADCINT8	ADCINT7	ADCINT6	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-10. ADC Interrupt Overflow Register (ADCINTOVF) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7-0	ADCINTx (x = 8 to 1)		ADC Interrupt Overflow Bits.
			Indicates if an overflow occurred when generating ADCINT pulses. If the respective ADCINTFLG bit is set and a selected additional EOC trigger is generated, then an overflow condition occurs.
		0	No ADC interrupt overflow event detected.
		1	ADC Interrupt overflow event detected.
			The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection.

Figure 10-22. ADC Interrupt Overflow Clear Register (ADCINTOVFCLR) (Address Offset 07h)

15						9	8
Reserved							
R-0							
7	6	5	4	3	2	1	0
ADCINT8	ADCINT7	ADCINT6	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0/W-1	R-0/W-1	R-0/W-1	R-0/W-1	R-0/W-1	R-0/W-1	R-0/W-1	R-0/W-1

LEGEND: R/W = Read/Write; R = Read only; R-0/W-1 =always read 0, write 1 to set; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-11. ADC Interrupt Overflow Clear Register (ADCINTOVFCLR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7-0	ADCINTx (x = 8 to 1)		ADC Interrupt Overflow Clear Bits.
		0	No action.
		1	Clears the respective overflow bit in the ADCINTOVF register. If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCINTOVF register, then hardware has priority and the ADCINTOVF bit will be set.

NOTE: The following Interrupt Select Registers are EALLOW protected.

Figure 10-23. Interrupt Select 1 And 2 Register (INTSEL1N2) (Address Offset 08h)

15	14	13	12	8
Reserved	INT2CONT	INT2E	INT2SEL	
R-0	R/W-0	R/W-0	R/W-0	

7	6	5	4	0
Reserved	INT1CONT	INT1E	INT1SEL	
R-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 10-24. Interrupt Select 3 And 4 Register (INTSEL3N4) (Address Offset 09h)

15	14	13	12	8
Reserved	INT4CONT	INT4E	INT4SEL	
R-0	R/W-0	R/W-0	R/W-0	

7	6	5	4	0
Reserved	INT3CONT	INT3E	INT3SEL	
R-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 10-25. Interrupt Select 5 And 6 Register (INTSEL5N6) (Address Offset 0Ah)

15	14	13	12	8
Reserved	INT6CONT	INT6E	INT6SEL	
R-0	R/W-0	R/W-0	R/W-0	

7	6	5	4	0
Reserved	INT5CONT	INT5E	INT5SEL	
R-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 10-26. Interrupt Select 7 And 8 Register (INTSEL7N8) (Address Offset 0Bh)

15	14	13	12	8
Reserved	INT8CONT	INT8E	INT8SEL	
R-0	R/W-0	R/W-0	R/W-0	

7	6	5	4	0
Reserved	INT7CONT	INT7E	INT7SEL	
R-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-27. Interrupt Select 9 And 10 Register (INTSEL9N10) (Address Offset 0Ch)

15				8
Reserved				
R-0				

7	6	5	4	0
Reserved	INT9CONT	INT9E	INT9SEL	
R-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-12. INTSELxNy Register Field Descriptions

Bit	Field	Value	Description
15	Reserved	0	Reserved
14	INTyCONT		ADCINTy Continuous Mode Enable
		0	No further ADCINTy pulses are generated until ADCINTy flag (in ADCINTFLG register) is cleared by user.
		1	ADCINTy pulses are generated whenever an EOC pulse is generated irrespective if the flag bit is cleared or not.
13	INTyE		ADCINTy Interrupt Enable
		0	ADCINTy is disabled.
		1	ADCINTy is enabled.
12-8	INTySEL		ADCINTy EOC Source Select
		00h	EOC0 is trigger for ADCINTy
		01h	EOC1 is trigger for ADCINTy
		02h	EOC2 is trigger for ADCINTy
		03h	EOC3 is trigger for ADCINTy
		04h	EOC4 is trigger for ADCINTy
		05h	EOC5 is trigger for ADCINTy
		06h	EOC6 is trigger for ADCINTy
		07h	EOC7 is trigger for ADCINTy
		08h	EOC8 is trigger for ADCINTy
		09h	EOC9 is trigger for ADCINTy
		0Ah	EOC10 is trigger for ADCINTy
		0Bh	EOC11 is trigger for ADCINTy
		0Ch	EOC12 is trigger for ADCINTy
		0Dh	EOC13 is trigger for ADCINTy
		0Eh	EOC14 is trigger for ADCINTy
		0Fh	EOC15 is trigger for ADCINTy
		1xh	Invalid value.
7	Reserved	0	Reserved
6	INTxCONT		ADCINTx Continuous Mode Enable.
		0	No further ADCINTx pulses are generated until ADCINTx flag (in ADCINTFLG register) is cleared by user.
		1	ADCINTx pulses are generated whenever an EOC pulse is generated irrespective if the flag bit is cleared or not.
5	INTxE		ADCINTx Interrupt Enable
		0	ADCINTx is disabled.
		1	ADCINTx is enabled .

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-12. INTSELxNy Register Field Descriptions (continued)

Bit	Field	Value	Description
4-0	INTxSEL		ADCINTx EOC Source Select
		00h	EOC0 is trigger for ADCINTx
		01h	EOC1 is trigger for ADCINTx
		02h	EOC2 is trigger for IADCNTx
		03h	EOC3 is trigger for ADCINTx
		04h	EOC4 is trigger for ADCINTx
		05h	EOC5 is trigger for ADCINTx
		06h	EOC6 is trigger for ADCINTx
		07h	EOC7 is trigger for ADCINTx
		08h	EOC8 is trigger for ADCINTx
		09h	EOC9 is trigger for ADCINTx
		0Ah	EOC10 is trigger for ADCINTx
		0Bh	EOC11 is trigger for ADCINTx
		0Ch	EOC12 is trigger for ADCINTx
		.0Dh	EOC13 is trigger for ADCINTx
		0Eh	EOC14 is trigger for ADCINTx
		0Fh	EOC15 is trigger for ADCINTx
		1xh	Invalid value.

10.3.11.4 ADC Priority Register

NOTE: The following SOC Priority Control Register is EALLOW protected.

Figure 10-28. ADC Start of Conversion Priority Control Register (SOCPRICTL)

15	14 11	10 5	4 0
ONESHOT	Reserved	RRPOINTER	SOCPRIORITY
R/W-0	R-0	R-20h	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-13. SOCPRICTL Register Field Descriptions

Bit	Field	Value	Description
15	ONESHOT	0	One shot mode disabled
		1	One shot mode enabled
14-11	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-13. SOCPRICTL Register Field Descriptions (continued)

Bit	Field	Value	Description
10-5	RRPOINTER		Round Robin Pointer. Holds the value of the last converted round robin SOCx to be used by the round robin scheme to determine order of conversions.
		00h	SOC0 was last round robin SOC to convert. SOC1 is highest round robin priority.
		01h	SOC1 was last round robin SOC to convert. SOC2 is highest round robin priority.
		02h	SOC2 was last round robin SOC to convert. SOC3 is highest round robin priority.
		03h	SOC3 was last round robin SOC to convert. SOC4 is highest round robin priority.
		04h	SOC4 was last round robin SOC to convert. SOC5 is highest round robin priority.
		05h	SOC5 was last round robin SOC to convert. SOC6 is highest round robin priority.
		06h	SOC6 was last round robin SOC to convert. SOC7 is highest round robin priority.
		07h	SOC7 was last round robin SOC to convert. SOC8 is highest round robin priority.
		08h	SOC8 was last round robin SOC to convert. SOC9 is highest round robin priority.
		09h	SOC9 was last round robin SOC to convert. SOC10 is highest round robin priority.
		0Ah	SOC10 was last round robin SOC to convert. SOC11 is highest round robin priority.
		0Bh	SOC11 was last round robin SOC to convert. SOC12 is highest round robin priority.
		0Ch	SOC12 was last round robin SOC to convert. SOC13 is highest round robin priority.
		0Dh	SOC13 was last round robin SOC to convert. SOC14 is highest round robin priority.
		0Eh	SOC14 was last round robin SOC to convert. SOC15 is highest round robin priority.
		0Fh	SOC15 was last round robin SOC to convert. SOC0 is highest round robin priority.
		1xh	Invalid value
		20h	Reset value to indicate no SOC has been converted. SOC0 is highest round robin priority. Set to this value when the device is reset, when the ADCCTL1.RESET bit is set, or when the SOCPRICTL register is written. In the latter case, if a conversion is currently in progress, it will complete and then the new priority will take effect.
		Others	Invalid selection.
4-0	SOCPRIORITY		SOC Priority.
			Determines the cutoff point for priority mode and round robin arbitration for SOCx
		00h	SOC priority is handled in round robin mode for all channels.
		01h	SOC0 is high priority, rest of channels are in round robin mode.
		02h	SOC0-SOC1 are high priority, SOC2-SOC15 are in round robin mode.
		03h	SOC0-SOC2 are high priority, SOC3-SOC15 are in round robin mode.
		04h	SOC0-SOC3 are high priority, SOC4-SOC15 are in round robin mode.
		05h	SOC0-SOC4 are high priority, SOC5-SOC15 are in round robin mode.
		06h	SOC0-SOC5 are high priority, SOC6-SOC15 are in round robin mode.
		07h	SOC0-SOC6 are high priority, SOC7-SOC15 are in round robin mode.
		08h	SOC0-SOC7 are high priority, SOC8-SOC15 are in round robin mode.
		09h	SOC0-SOC8 are high priority, SOC9-SOC15 are in round robin mode.
		0Ah	SOC0-SOC9 are high priority, SOC10-SOC15 are in round robin mode.
		0Bh	SOC0-SOC10 are high priority, SOC11-SOC15 are in round robin mode.
		0Ch	SOC0-SOC11 are high priority, SOC12-SOC15 are in round robin mode.
		0Dh	SOC0-SOC12 are high priority, SOC13-SOC15 are in round robin mode.
		0Eh	SOC0-SOC13 are high priority, SOC14-SOC15 are in round robin mode.
		0Fh	SOC0-SOC14 are high priority, SOC15 is in round robin mode.
		10h	All SOCs are in high priority mode, arbitrated by SOC number
		Others	Invalid selection.

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.11.5 ADC SOC Registers

NOTE: The following ADC Sample Mode Register is EALLOW protected.

Figure 10-29. ADC Sample Mode Register (ADCSAMPLEMODE) (Address Offset 12h)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
SIMULEN14	SIMULEN12	SIMULEN10	SIMULEN8	SIMULEN6	SIMULEN4	SIMULEN2	SIMULEN0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-14. ADC Sample Mode Register (ADCSAMPLEMODE) Field Descriptions

Bit	Field	Value	Description
15:8	Reserved	0	Reserved
7	SIMULEN14		Simultaneous sampling enable for SOC14/SOC15. Couples SOC14 and SOC15 in simultaneous sampling mode. See Section 10.3.6 for details. This bit should not be set when the ADC is actively converting SOC14 or SOC15.
		0	Single sample mode set for SOC14 and SOC15. All bits of CHSEL field define channel to be converted. EOC14 associated with SOC14. EOC15 associated with SOC15. SOC14's result placed in ADCRESULT14 register. SOC15's result placed in ADCRESULT15.
		1	Simultaneous sample for SOC14 and SOC15. Lowest three bits of CHSEL field define the pair of channels to be converted. EOC14 and EOC15 associated with SOC14 and SOC15 pair. SOC14's and SOC15's results will be placed in ADCRESULT14 and ADCRESULT15 registers, respectively.
6	SIMULEN12		Simultaneous sampling enable for SOC12/SOC13. Couples SOC12 and SOC13 in simultaneous sampling mode. See Section 10.3.6 for details. This bit should not be set when the ADC is actively converting SOC12 or SOC13.
		0	Single sample mode set for SOC12 and SOC13. All bits of CHSEL field define channel to be converted. EOC12 associated with SOC12. EOC13 associated with SOC13. SOC12's result placed in ADCRESULT12 register. SOC13's result placed in ADCRESULT13.
		1	Simultaneous sample for SOC12 and SOC13. Lowest three bits of CHSEL field define the pair of channels to be converted. EOC12 and EOC13 associated with SOC12 and SOC13 pair. SOC12's and SOC13's results will be placed in ADCRESULT12 and ADCRESULT13 registers, respectively.
5	SIMULEN10		Simultaneous sampling enable for SOC10/SOC11. Couples SOC10 and SOC11 in simultaneous sampling mode. See Section 10.3.6 for details. This bit should not be set when the ADC is actively converting SOC10 or SOC11.
		0	Single sample mode set for SOC10 and SOC11. All bits of CHSEL field define channel to be converted. EOC10 associated with SOC10. EOC11 associated with SOC11. SOC10's result placed in ADCRESULT10 register. SOC11's result placed in ADCRESULT11.
		1	Simultaneous sample for SOC10 and SOC11. Lowest three bits of CHSEL field define the pair of channels to be converted. EOC10 and EOC11 associated with SOC10 and SOC11 pair. SOC10's and SOC11's results will be placed in ADCRESULT10 and ADCRESULT11 registers, respectively.
4	SIMULEN8		Simultaneous sampling enable for SOC8/SOC9. Couples SOC8 and SOC9 in simultaneous sampling mode. See Section 10.3.6 for details. . This bit should not be set when the ADC is actively converting SOC8 or SOC9.
		0	Single sample mode set for SOC8 and SOC9. All bits of CHSEL field define channel to be converted. EOC8 associated with SOC8. EOC9 associated with SOC9. SOC8's result placed in ADCRESULT8 register. SOC9's result placed in ADCRESULT9.
		1	Simultaneous sample for SOC8 and SOC9. Lowest three bits of CHSEL field define the pair of channels to be converted. EOC8 and EOC9 associated with SOC8 and SOC9 pair. SOC8's and SOC9's results will be placed in ADCRESULT8 and ADCRESULT9 registers, respectively.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-14. ADC Sample Mode Register (ADCSAMPLEMODE) Field Descriptions (continued)

Bit	Field	Value	Description
3	SIMULEN6		Simultaneous sampling enable for SOC6/SOC7. Couples SOC6 and SOC7 in simultaneous sampling mode. See Section 10.3.6 for details. This bit should not be set when the ADC is actively converting SOC6 or SOC7.
		0	Single sample mode set for SOC6 and SOC7. All bits of CHSEL field define channel to be converted. EOC6 associated with SOC6. EOC7 associated with SOC7. SOC6's result placed in ADCRESULT6 register. SOC7's result placed in ADCRESULT7.
		1	Simultaneous sample for SOC6 and SOC7. Lowest three bits of CHSEL field define the pair of channels to be converted. EOC6 and EOC7 associated with SOC6 and SOC7 pair. SOC6's and SOC7's results will be placed in ADCRESULT6 and ADCRESULT7 registers, respectively.
2	SIMULEN4		Simultaneous sampling enable for SOC4/SOC5. Couples SOC4 and SOC5 in simultaneous sampling mode. See Section 10.3.6 for details. This bit should not be set when the ADC is actively converting SOC4 or SOC5.
		0	Single sample mode set for SOC4 and SOC5. All bits of CHSEL field define channel to be converted. EOC4 associated with SOC4. EOC5 associated with SOC5. SOC4's result placed in ADCRESULT4 register. SOC5's result placed in ADCRESULT5.
		1	Simultaneous sample for SOC4 and SOC5. Lowest three bits of CHSEL field define the pair of channels to be converted. EOC4 and EOC5 associated with SOC4 and SOC5 pair. SOC4's and SOC5's results will be placed in ADCRESULT4 and ADCRESULT5 registers, respectively.
1	SIMULEN2		Simultaneous sampling enable for SOC2/SOC3. Couples SOC2 and SOC3 in simultaneous sampling mode. See Section 10.3.6 for details. This bit should not be set when the ADC is actively converting SOC2 or SOC3.
		0	Single sample mode set for SOC2 and SOC3. All bits of CHSEL field define channel to be converted. EOC2 associated with SOC2. EOC3 associated with SOC3. SOC2's result placed in ADCRESULT2 register. SOC3's result placed in ADCRESULT3.
		1	Simultaneous sample for SOC2 and SOC3. Lowest three bits of CHSEL field define the pair of channels to be converted. EOC2 and EOC3 associated with SOC2 and SOC3 pair. SOC2's and SOC3's results will be placed in ADCRESULT2 and ADCRESULT3 registers, respectively.
0	SIMULEN0		Simultaneous sampling enable for SOC0/SOC1. Couples SOC0 and SOC1 in simultaneous sampling mode. See Section 10.3.6 for details. This bit should not be set when the ADC is actively converting SOC0 or SOC1.
		0	Single sample mode set for SOC0 and SOC1. All bits of CHSEL field define channel to be converted. EOC0 associated with SOC0. EOC1 associated with SOC1. SOC0's result placed in ADCRESULT0 register. SOC1's result placed in ADCRESULT1.
		1	Simultaneous sample for SOC0 and SOC1. Lowest three bits of CHSEL field define the pair of channels to be converted. EOC0 and EOC1 associated with SOC0 and SOC1 pair. SOC0's and SOC1's results will be placed in ADCRESULT0 and ADCRESULT1 registers, respectively.

NOTE: The following ADC Interrupt SOC Select Registers are EALLOW protected.

Figure 10-30. ADC Interrupt Trigger SOC Select 1 Register (ADCINTSOCSEL1) (Address Offset 14h)

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-15. ADC Interrupt Trigger SOC Select 1 Register (ADCINTSOCSEL1) Register Field Descriptions

Bit	Field	Value	Description
15--0	SOCx (x = 7 to 0)		SOCx ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOCx. This field overrides the TRIGSEL field in the ADCSOCxCTL register.
		00	No ADCINT will trigger SOCx. TRIGSEL field determines SOCx trigger.
		01	ADCINT1 will trigger SOCx. TRIGSEL field is ignored.
		10	ADCINT2 will trigger SOCx. TRIGSEL field is ignored.
		11	Invalid selection.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-31. ADC Interrupt Trigger SOC Select 2 Register (ADCINTSOCSEL2) (Address Offset 15h)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SOC15		SOC14		SOC13		SOC12		SOC11		SOC10		SOC9		SOC8	
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-16. ADC Interrupt Trigger SOC Select 2 Register (ADCINTSOCSEL2) Field Descriptions

Bit	Field	Value	Description
15-0	SOCx (x = 15 to 8)		SOCx ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOCx. This field overrides the TRIGSEL field in the ADCSOCxCTL register.
		00	No ADCINT will trigger SOCx. TRIGSEL field determines SOCx trigger.
		01	ADCINT1 will trigger SOCx. TRIGSEL field is ignored.
		10	ADCINT2 will trigger SOCx. TRIGSEL field is ignored.
		11	Invalid selection.

Figure 10-32. ADC SOC Flag 1 Register (ADCSOCFLG1) (Address Offset 18h)

15	14	13	12	11	10	9	8
SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-17. ADC SOC Flag 1 Register (ADCSOCFLG1) Field Descriptions

Bit	Field	Value	Description
15-0	SOCx (x = 15 to 0)		SOCx Start of Conversion Flag. Indicates the state of individual SOC conversions.
		0	No sample pending for SOCx.
		1	Trigger has been received and sample is pending for SOCx.
			The bit will be automatically cleared when the respective SOCx conversion is started. If contention exists where this bit receives both a request to set **and** a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.

Figure 10-33. ADC SOC Force 1 Register (ADCSOCFRC1) (Address Offset 1Ah)

15	14	13	12	11	10	9	8
SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-18. ADC SOC Force 1 Register (ADCSOCFRC1) Field Descriptions

Bit	Field	Value	Description
15-0	SOCx (x = 15 to 0)		SOCx Force Start of Conversion Flag. Writing a 1 will force to 1 the respective SOCx flag bit in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored.
		0	No action.
		1	Force SOCx flag bit to 1. This will cause a conversion to start once priority is given to SOCx.
			If software tries to set this bit on the same clock cycle that hardware tries to clear the SOCx bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.

Figure 10-34. ADC SOC Overflow 1 Register (ADCSOCOVF1) (Address Offset 1Ch)

15	14	13	12	11	10	9	8
SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-19. ADC SOC Overflow 1 Register (ADCSOCOVF1) Field Descriptions

Bit	Field	Value	Description
15-0	SOCx (x = 15 to 0)		SOCx Start of Conversion Overflow Flag. Indicates an SOCx event was generated while an existing SOCx event was already pending.
		0	No SOCx event overflow
		1	SOCx event overflow
			An overflow condition does not stop SOCx events from being processed. It simply is an indication that a trigger was missed

Figure 10-35. ADC SOC Overflow Clear 1 Register (ADCSOCOVFCLR1) (Address Offset 1Eh)

15	14	13	12	11	10	9	8
SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-20. ADC SOC Overflow Clear 1 Register (ADCSOCOVFCLR1) Field Descriptions

Bit	Field	Value	Description
15-0	SOCx (x = 15 to 0)		SOCx Clear Start of Conversion Overflow Flag. Writing a 1 will clear the respective SOCx overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored.
		0	No action.
		1	Clear SOCx overflow flag.
			If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set.

NOTE: The following ADC SOC0 - SOC15 Control Registers are EALLOW protected.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-36. ADC SOC0 - SOC15 Control Registers (ADCSOCxCTL) (Address Offset 20h - 2Fh)

15 11	10	9 6	5 0
TRIGSEL	Reserved	CHSEL	ACQPS
R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-21. ADC SOC0 - SOC15 Control Registers (ADCSOCxCTL) Register Field Descriptions

Bit	Field	Value	Description
15-11	TRIGSEL		SOCx Trigger Source Select. Configures which trigger will set the respective SOCx flag in the ADCSOCFLG1 register to intiate a conversion to start once priority is given to SOCx. This setting can be overridden by the respective SOCx field in the ADCINTSOCSEL1 or ADCINTSOCSEL2 register.
		00h	SOCx Trigger Disable (Default)
		05h	ADCTRIG1 – ADC Trigger 1
		06h	ADCTRIG2 – ADC Trigger 2
		07h	ADCTRIG3 – ADC Trigger 3
		08h	ADCTRIG4 – ADC Trigger 4
		09h	ADCTRIG5 – ADC Trigger 5
		0Ah	ADCTRIG6 – ADC Trigger 6
		0Bh	ADCTRIG7 – ADC Trigger 7
		0Ch	ADCTRIG8 – ADC Trigger 8
		Others	Invalid selection.
10	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-21. ADC SOC0 - SOC15 Control Registers (ADCSOCxCTL) Register Field Descriptions (continued)

Bit	Field	Value	Description
9-6	CHSEL		SOCx Channel Select. Selects the channel to be converted when SOCx is received by the ADC.
			Sequential Sampling Mode (SIMULENx = 0):
		0h	ADCINA0
		1h	ADCINA1
		2h	ADCINA2
		3h	ADCINA3
		4h	ADCINA4
		5h	ADCINA5
		6h	ADCINA6
		7h	ADCINA7
		8h	ADCINB0
		9h	ADCINB1
		Ah	ADCINB2
		Bh	ADCINB3
		Ch	ADCINB4
		Dh	ADCINB5
		Eh	ADCINB6
		Fh	ADCINB7
			Simultaneous Sampling Mode (SIMULENx = 1):
		0h	ADCINA0/ADCINB0 pair
		1h	ADCINA1/ADCINB1 pair
		2h	ADCINA2/ADCINB2 pair
		3h	ADCINA3/ADCINB3 pair
		4h	ADCINA4/ADCINB4 pair
		5h	ADCINA5/ADCINB5 pair
		6h	ADCINA6/ADCINB6 pair
		7h	ADCINA7/ADCINB7 pair
		8h	Invalid selection.
		9h	Invalid selection.
		Ah	Invalid selection.
		Bh	Invalid selection.
		Ch	Invalid selection.
		Dh	Invalid selection.
		Eh	Invalid selection.
		Fh	Invalid selection.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-21. ADC SOC0 - SOC15 Control Registers (ADCSOCxCTL) Register Field Descriptions (continued)

Bit	Field	Value	Description
5-0	ACQPS		SOCx Acquisition Prescale. Controls the sample and hold window for SOCx. Minimum value allowed is 6.
		00h	Invalid selection.
		01h	Invalid selection.
		02h	Invalid selection.
		03h	Invalid selection.
		04h	Invalid selection.
		05h	Invalid selection.
		06h	Sample window is 7 cycles long (6 + 1 clock cycles).
		07h	Sample window is 8 cycles long (7 + 1 clock cycles).
		08h	Sample window is 9 cycles long (8 + 1 clock cycles).
		09h	Sample window is 10 cycles long (9 + 1 clock cycles).
		...	...
		3Fh	Sample window is 64 cycles long (63 + 1 clock cycles).
Other invalid selections: 10h, 11h, 12h, 13h, 14h, 1Dh, 1Eh, 1Fh, 20h, 21h, 2Ah, 2Bh, 2Ch, 2Dh, 2Eh, 37h, 38h, 39h, 3Ah, 3Bh			

NOTE: Also configure the TRIGxSEL registers for which trigger source will be tied to each ADC trigger.

10.3.11.6 ADC Calibration Registers

NOTE: The following ADC Calibration Registers are EALLOW protected.

Figure 10-37. ADC Reference/Gain Trim Register (ADCREFTRIM) (Address Offset 40h)

15 13	12 8	7 4	3 0
Reserved	EXTREF_FINE_TRIM	BG_COARSE_TRIM	BG_FINE_TRIM
R-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-22. ADC Reference/Gain Trim Register (ADCREFTRIM) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved		Reserved
12-8	EXTREF_FINE_TRIM		ADC External reference Fine Trim. These bits should not be modified after device boot code loads them with the factory trim setting.
7-4	BG_COARSE_TRIM		ADC Internal Bandgap Fine Trim. These bits should not be modified after device boot code loads them with the factory trim setting.
3-0	BG_FINE_TRIM		ADC Internal Bandgap Coarse Trim. A maximum value of 30 is supported. These bits should not be modified after device boot code loads them with the factory trim setting.

Figure 10-38. ADC Offset Trim Register (ADCOFFTRIM) (Address Offset 41h)

15 9	8 0
Reserved	OFFTRIM
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-23. ADC Offset Trim Register (ADCOFFTRIM) Field Descriptions

Bit	Field	Value	Description
15-9	Reserved		Reserved
8-0	OFFTRIM		ADC Offset Trim. 2's complement of ADC offset. Range is -256 to +255. These bits are loaded by device boot code with a factory trim setting. Modification of this default setting can be made to correct any board induced offset.

10.3.11.7 ADC Revision Register

Figure 10-39. ADC Revision Register (ADCREV) (Address Offset 4Fh)

15	8
REV	
R-x	

7	0
TYPE	
R-3h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-24. ADC Revision Register (ADCREV) Field Descriptions

Bit	Field	Value	Description
15-8	REV		ADC Revision. To allow documentation of differences between revisions. First version is labeled as 00h.
7-0	TYPE	3	ADC Type. Always set to 3 for this type ADC

10.3.11.8 ADC Result Registers

The ADC Result Registers are found in Peripheral Frame 0 (PF0). In the header files, the ADCRESULTx registers are located in the AdcxResult register file, not AdcxRegs. The ADC Result Registers can also be accessed by the M3 CPU.

Figure 10-40. ADC RESULT0 - RESULT15 Registers (ADCRESULTx) (PF1 Block Address Offset 00h - 0Fh)

15	12	11	0
Reserved		RESULT	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-25. ADC RESULT0 - ADCRESULT15 Registers (ADCRESULTx) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved		Reserved
11-0	RESULT		12-bit right-justified ADC result Sequential Sampling Mode (SIMULENx = 0): After the ADC completes a conversion of an SOCx, the digital result is placed in the corresponding ADCRESULTx register. For example, if SOC4 is configured to sample ADCINA1, the completed result of that conversion will be placed in ADCRESULT4. Simultaneous Sampling Mode (SIMULENx = 1): After the ADC completes a conversion of a channel pair, the digital results are found in the corresponding ADCRESULTx and ADCRESULTx+1 registers (assuming x is even). For example, for SOC4, the completed results of those conversions will be placed in ADCRESULT4 and ADCRESULT5. See 1.11 for timings of when this register is written.

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.12 Analog Subsystem Control Registers

This section contains the analog subsystem registers and bit definitions.

Table 10-26. Analog Subsystem Control Registers (AnalogSysctrlReg)

Register Name	Address Offset	Size (x16)	Description
INTOVF	10h	1	ADC Interrupt Overflow Detect
INTOVFCLR	11h	1	ADC Interrupt Overflow Clear
CLOCK	40h	1	Control System: Lock Register(1)
CCIBSTATUS	41h	1	Control System: ACIB Status Register
CCLKCTL	42h	1	Control System: Clock Control(1)
TRIGOVF	50h	1	ADC Trigger Overflow Detect
TRIGOVFCLR	51h	1	ADC Trigger Overflow Clear
TRIG1SEL-TRIG8SEL	70h - 77h	1	ADC Trigger 1-8 Input Select(1)

(1) This register is EALLOW protected.

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.12.1 ADC Interrupt Overflow Detect Register (INTOVF)

Figure 10-41. ADC Interrupt Overflow Detect Register (INTOVF)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
ADCINT8	ADCINT7	ADCINT6	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-27. ADC Interrupt Overflow Detect Register (INTOVF) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	ADCINT8		ADCINT8 Overflow Flag Status
		0	No overflow.
		1	Overflow detected.
6	ADCINT7		ADCINT7 Overflow Flag Status
		0	No overflow.
		1	Overflow detected.
5	ADCINT6		ADCINT6 Overflow Flag Status
		0	No overflow.
		1	Overflow detected.
4	ADCINT5		ADCINT5 Overflow Flag Status
		0	No overflow.
		1	Overflow detected.
3	ADCINT4		ADCINT4 Overflow Flag Status
		0	No overflow.
		1	Overflow detected.
2	ADCINT3		ADCINT3 Overflow Flag Status
		0	No overflow.
		1	Overflow detected.
1	ADCINT2		ADCINT2 Overflow Flag Status
		0	No overflow.
		1	Overflow detected.
0	ADCINT1		ADCINT1 Overflow Flag Status
		0	No overflow.
		1	Overflow detected.

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.12.2 ADC Interrupt Overflow Clear Register (INTOVFCLR)

Figure 10-42. ADC Interrupt Overflow Clear Register (INTOVFCLR)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
ADCINT8	ADCINT7	ADCINT6	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-28. ADC Interrupt Overflow Clear Register (INTOVFCLR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	ADCINT8		ADCINT8 Overflow Flag Clear
		0	No action.
		1	Clears overflow flag.
6	ADCINT7		ADCINT7 Overflow Flag Clear
		0	No action.
		1	Clears overflow flag.
5	ADCINT6		ADCINT6 Overflow Flag Clear
		0	No action.
		1	Clears overflow flag.
4	ADCINT5		ADCINT5 Overflow Flag Clear
		0	No action.
		1	Clears overflow flag.
3	ADCINT4		ADCINT4 Overflow Flag Clear
		0	No action.
		1	Clears overflow flag.
2	ADCINT3		ADCINT3 Overflow Flag Clear
		0	No action.
		1	Clears overflow flag.
1	ADCINT2		ADCINT2 Overflow Flag Clear
		0	No action.
		1	Clears overflow flag.
0	ADCINT1		ADCINT1 Overflow Flag Clear
		0	No action.
		1	Clears overflow flag.

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.12.3 Control System: Lock Register (CLOCK)

NOTE: This Analog Subsystem Control Register is EALLOW protected.

Figure 10-43. Control System: Lock Register (CLOCK)

15							8
PSWD							
R-0/W							

7					2	1	0
Reserved						CCLKCTL	Reserved
R-0						R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-29. Control System: Lock Register (CLOCK) Field Descriptions

Bit	Field	Value	Description
15-8	PSWD	0-FFh	Write protection password The CCLKCTL protection bit can only be written if a proper password is written to PSWD simultaneously. The password is 1Bh.
7-2	Reserved	0	Reserved
1	CCLKCTL		Control System: CLKCTL Register Write Disable. This bit, if written simultaneously with the correct PSWD value, will enable write protection for the CLKDIV bits in the CCLKCTL register. Write protection can only be disabled by a system reset.
0	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.12.4 Control System: ACIB Status Register (CCIBSTATUS)

Figure 10-44. Control System: ACIB Status Register (CCIBSTATUS)

15				8
ASYSCLKCNT				
R-0				
7	**3**	**2**	**1**	**0**
Reserved		INT	READY	Reserved
R-0		R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-30. Control System: ACIB Status Register (CCIBSTATUS) Field Descriptions

Bit	Field	Value	Description
15-8	ASYSCLKCNT	0-FFh	8-bit ACIB Bus Clock Counter This is a free running counter clocked by the ACIB clock. This counter indicates if the ACIB clock is present.
7-3	Reserved	0	Reserved
2	INT		INT signal state Reads of this bit will give the current state of the INT signal.
1	READY		READY signal state Reads of this bit will give the current state of the READY signal.
0	APGOODSTS		Analog Subsystem Power Good Status
		0	Power not present
		1	Power present

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.12.5 Control System: Clock Control Register (CCLKCTL)

NOTE: This Analog Subsystem Control Register is EALLOW protected.

Figure 10-45. Control System: Clock Control Register (CCLKCTL)

15								8
Reserved								
R-0								
7				3	2			0
Reserved					CLKDIV			
R-0					R/W-4h			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-31. Control System: Clock Control Register (CCLKCTL) Field Descriptions

Bit	Field	Value	Description
15-3	Reserved	0	Reserved
2-0	CLKDIV		Analog Subsystem Source Clock Divisor Selection.
		0	Clock is turned off.
		001	Divide by /1 mode
		010	Divide by /2 mode
		011	Divide by /4 mode
		100	Divide by /8 mode (Default)
		101	Reserved
		110	Reserved
		111	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.12.6 ADC Start of Conversion Trigger Overflow Detect Register (TRIGOVF)

Figure 10-46. ADC Start of Conversion Trigger Overflow Detect Register (TRIGOVF)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
TRIG8	TRIG7	TRIG6	TRIG5	TRIG4	TRIG3	TRIG2	TRIG1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-32. ADC Start of Conversion Trigger Overflow Detect Register (TRIGOVF) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	TRIG8		Indicates if overflow occurred on respective ADC trigger.
		0	No overflow.
		1	Overflow detected.
6	TRIG7		Indicates if overflow occurred on respective ADC trigger.
		0	No overflow.
		1	Overflow detected.
5	TRIG6		Indicates if overflow occurred on respective ADC trigger.
		0	No overflow.
		1	Overflow detected.
4	TRIG5		Indicates if overflow occurred on respective ADC trigger.
		0	No overflow.
		1	Overflow detected.
3	TRIG4		Indicates if overflow occurred on respective ADC trigger.
		0	No overflow.
		1	Overflow detected.
2	TRIG3		Indicates if overflow occurred on respective ADC trigger.
		0	No overflow.
		1	Overflow detected.
1	TRIG2		Indicates if overflow occurred on respective ADC trigger.
		0	No overflow.
		1	Overflow detected.
0	TRIG1		Indicates if overflow occurred on respective ADC trigger.
		0	No overflow.
		1	Overflow detected.

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.12.7 ADC Start of Conversion Trigger Overflow Flag Clear Register (TRIGOVFCLR)

Figure 10-47. ADC Start of Conversion Trigger Overflow Flag Clear Register (TRIGOVFCLR)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
TRIG8	TRIG7	TRIG6	TRIG5	TRIG4	TRIG3	TRIG2	TRIG1
R-0/W-1	R-0/W-1	R-0/W-1	R-0/W-1	R-0/W-1	R-0/W-1	R-0/W-1	R-0/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-33. ADC Start of Conversion Trigger Overflow Flag Clear Register (TRIGOVFCLR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	TRIG8		Clears ADC trigger overflow flag in TRIGOVF register.
		0	No action.
		1	Clears overflow flag.
6	TRIG7		Clears ADC trigger overflow flag in TRIGOVF register.
		0	No action.
		1	Clears overflow flag.
5	TRIG6		Clears ADC trigger overflow flag in TRIGOVF register.
		0	No action.
		1	Clears overflow flag.
4	TRIG5		Clears ADC trigger overflow flag in TRIGOVF register.
		0	No action.
		1	Clears overflow flag.
3	TRIG4		Clears ADC trigger overflow flag in TRIGOVF register.
		0	No action.
		1	Clears overflow flag.
2	TRIG3		Clears ADC trigger overflow flag in TRIGOVF register.
		0	No action.
		1	Clears overflow flag.
1	TRIG2		Clears ADC trigger overflow flag in TRIGOVF register.
		0	No action.
		1	Clears overflow flag.
0	TRIG1		Clears ADC trigger overflow flag in TRIGOVF register.
		0	No action.
		1	Clears overflow flag.

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.12.8 ADC Start of Conversion Trigx Input Select Register (TRIGxSEL)

Figure 10-48. ADC Start of Conversion Trigx Input Select Register (TRIGxSEL)

15	8
Reserved	
R-0	

7	5	4	0
Reserved		TRIGxSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

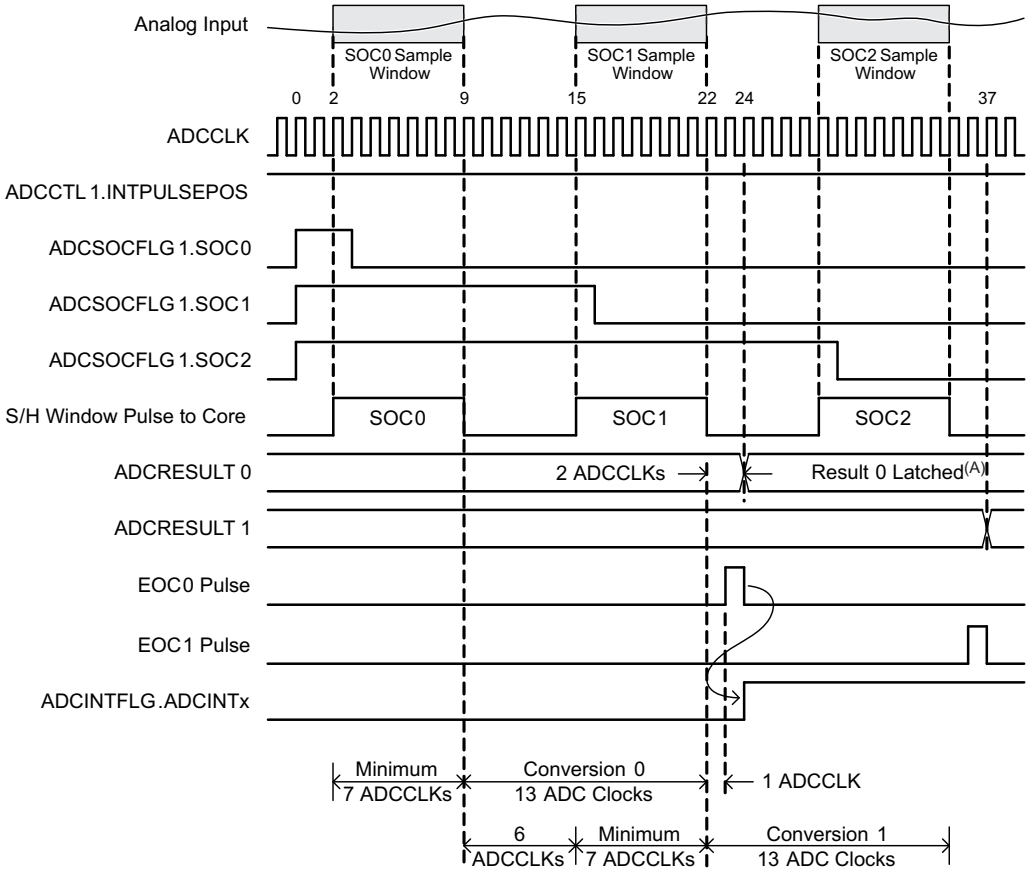
Table 10-34. ADC Start of Conversion Trigx Input Select Register (TRIGxSEL) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved
4-0	TRIGxSEL		Analog Trigger (TRIGx) MUX Input Select
		00000	No selection (Disabled)
		00001	TINT0 (CPU Timer 0)
		00010	TINT1 (CPU Timer 1)
		00011	TINT2 (CPU Timer 2)
		00100	ADCEXTTRIG (C28 GPIO MUX)
		00101	EPWM1SOCA (EPWM1)
		00110	EPWM1SOCB (EPWM1)
		00111	EPWM1SYNC (EPWM1)
		01000	EPWM2SOCA (EPWM2)
		01001	EPWM2SOCB (EPWM2)
		01010	EPWM2SYNC (EPWM2)
		01011	EPWM3SOCA (EPWM3)
		01100	EPWM3SOCB (EPWM3)
		01101	EPWM3SYNC (EPWM3)
		01110	EPWM4SOCA (EPWM4)
		01111	EPWM4SOCB (EPWM4)
		10000	EPWM4SYNC (EPWM4)
		10001	EPWM5SOCA (EPWM5)
		10010	EPWM5SOCB (EPWM5)
		10011	EPWM5SYNC (EPWM5)
		10100	EPWM6SOCA (EPWM6)
		10101	EPWM6SOCB (EPWM6)
		10110	EPWM6SYNC (EPWM6)
		10111	EPWM7SOCA (EPWM7)
		11000	EPWM7SOCB (EPWM7)
		11001	EPWM7SYNC (EPWM7)
		11010	EPWM8SOCA (EPWM8)
		11011	EPWM8SOCB (EPWM8)
		11100	EPWM8SYNC (EPWM8)
		11101	EPWM9SOCA (EPWM9)
		11110	EPWM9SOCB (EPWM9)
		11111	EPWM9SYNC (EPWM9)

Copyright © 2012–2019, Texas Instruments Incorporated

10.3.13 ADC Timings

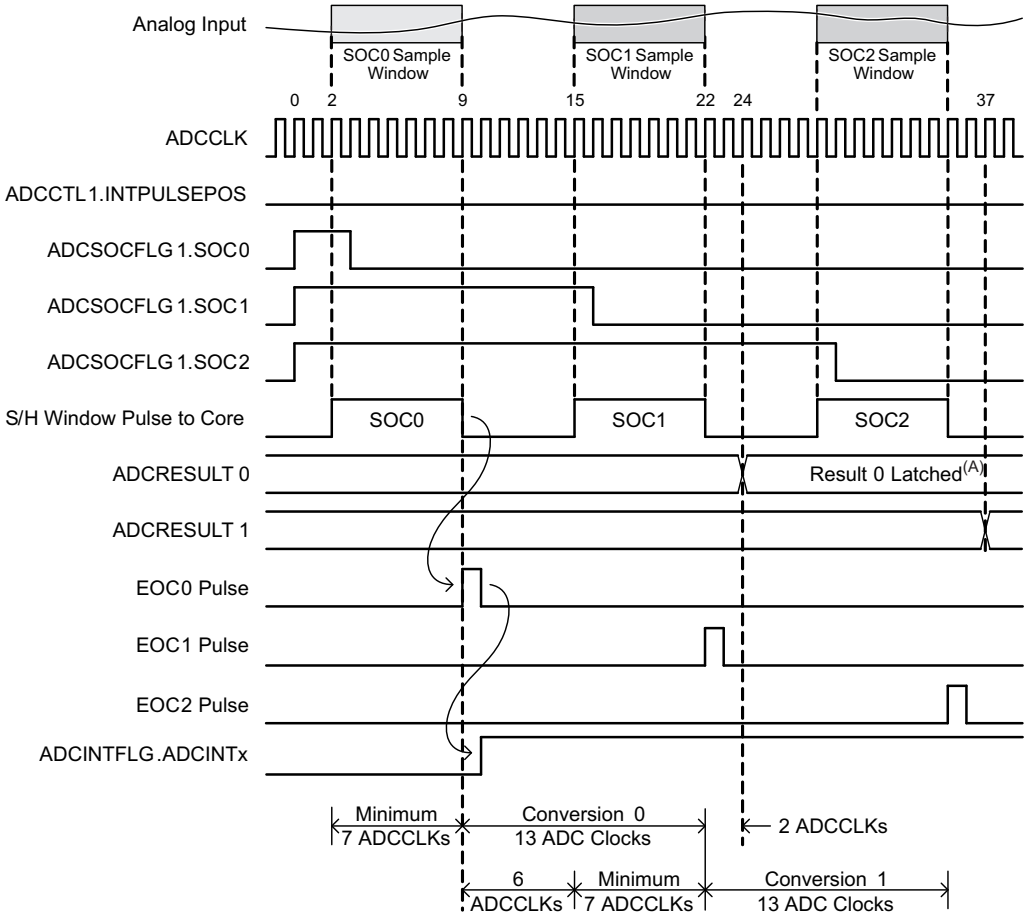
Figure 10-49. Timing Example For Sequential Mode / Late Interrupt Pulse



A Result 0 latched on this cycle does not include the additional cycles required for the C28x and M3 subsystems to read the ADC result registers using the ACIB.

Copyright © 2012–2019, Texas Instruments Incorporated

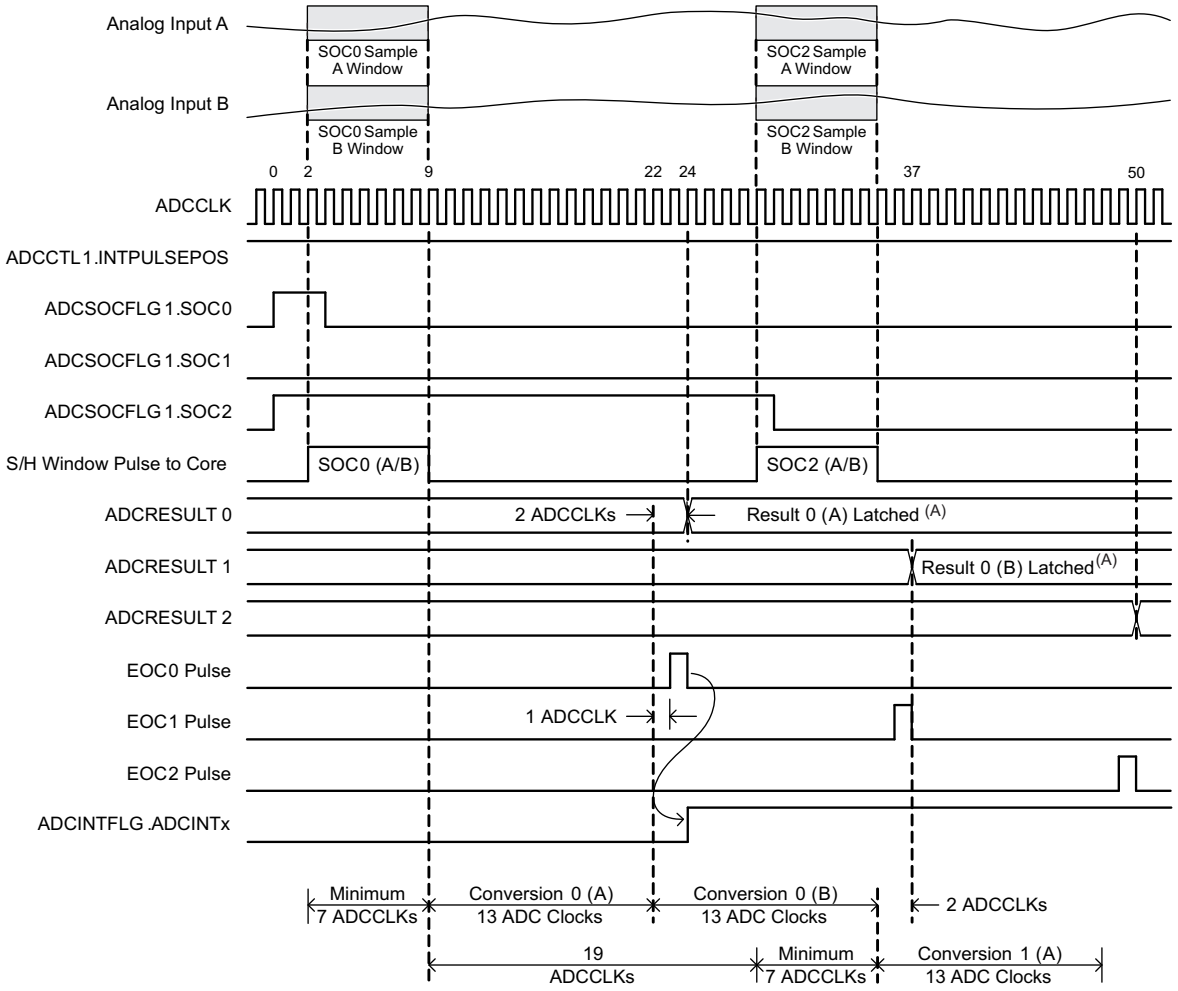
Figure 10-50. Timing Example For Sequential Mode / Early Interrupt Pulse



A Result 0 latched on this cycle does not include the additional cycles required for the C28x and M3 subsystems to read the ADC result registers using the ACIB.

Copyright © 2012–2019, Texas Instruments Incorporated

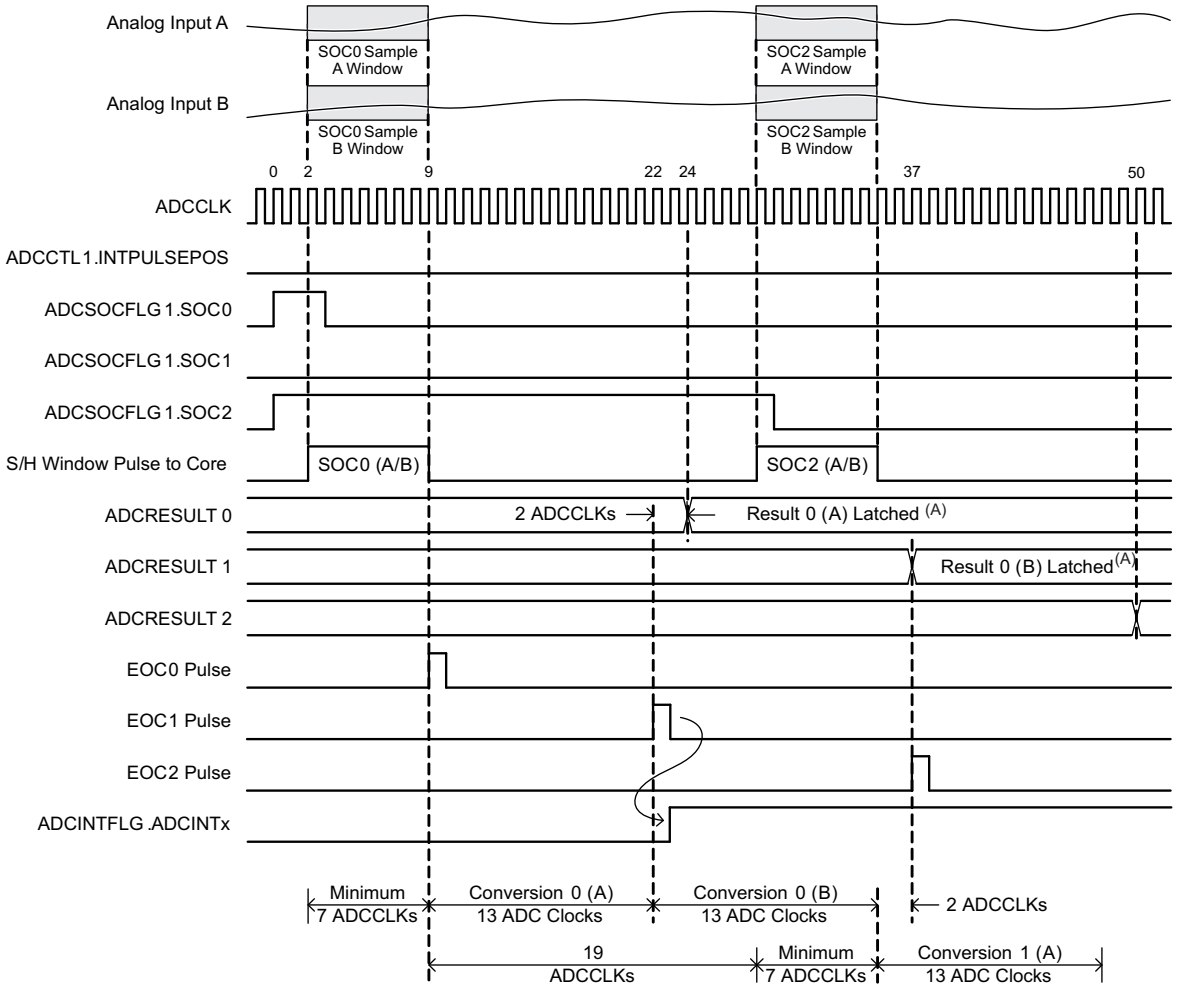
Figure 10-51. Timing Example For Simultaneous Mode / Late Interrupt Pulse



A Result 0 (A) and Result 0 (B) latched on their respective cycles does not include the additional cycles required for the C28x and M3 subsystems to read the ADC result registers using the ACIB.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 10-52. Timing Example For Simultaneous Mode / Early Interrupt Pulse



A Result 0 (A) and Result 0 (B) latched on their respective cycles does not include the additional cycles required for the C28x and M3 subsystems to read the ADC result registers using the ACIB.

10.4 Comparator Block

The comparator module described in this reference guide is a true analog voltage comparator in the VDDA domain. The analog portion of the block include the comparator, its inputs and outputs, and the internal DAC reference. The digital circuits, referred to as the wrapper in this document, include the DAC controls, interface to other on-chip logic, output qualification block, and the control signals.

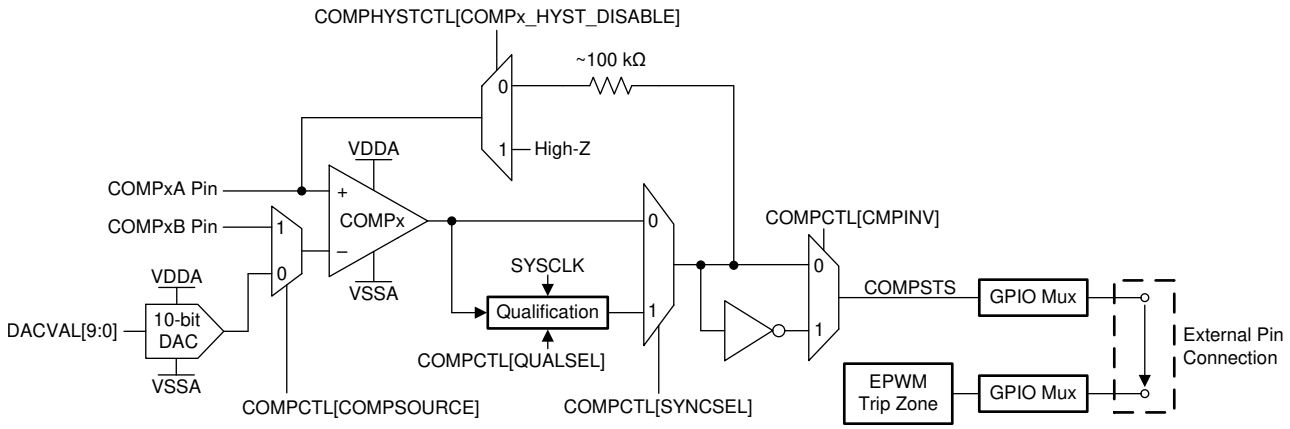
10.4.1 Features

The comparator block can accommodate two external analog inputs or one external analog input using the internal DAC reference for the other input. The output of the comparator can be passed asynchronously or qualified and synchronized to the system clock period. The comparator output can be externally connected to a GPIO in order to connect to an ePWM Trip Zone module.

Copyright © 2012–2019, Texas Instruments Incorporated

10.4.2 Comparator Block Diagram

Figure 10-53. Comparator Block Diagram



NOTE: Comparator hysteresis feedback is enabled by default and may interfere with high impedance input signals.

10.4.3 Comparator Function

The comparator in each comparator block is an analog comparator module, and as such its output is asynchronous to the system clock. The truth table for the comparator is shown in Table 10-35.

Figure 10-54. Comparator

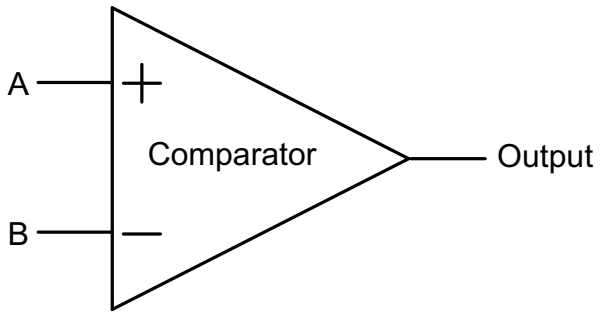


Table 10-35. Comparator Truth Table

Voltages	Output
Voltage A > Voltage B	1
Voltage B > Voltage A	0

There is no definition for the condition Voltage A = Voltage B since there is hysteresis in the response of the comparator output. Refer to the device datasheet for the value of this hysteresis. This also limits the sensitivity of the comparator output to noise on the input voltages.

The output state of the comparator, after qualification, is reflected by the COMPSTS bit in the COMPSTS register. Since this bit is part of the wrapper, clocks must be enabled to the comparator block for the COMPSTS bit to actively show the comparator state.

10.4.4 DAC Reference

Each comparator block contains a 10-bit voltage DAC reference that can used to supply the inverting input (B side input) of the comparator. The voltage output of the DAC is controlled by the DACVAL bit field in the DACVAL register. The output of the DAC is given by the equation:

Copyright © 2012–2019, Texas Instruments Incorporated

$$V = \frac{DACVAL * (VDDA-VSSA)}{1023}$$

Since the DAC is also in the analog domain it does not require a clock to maintain its voltage output. A clock is required, however, to modify the digital inputs that control the DAC.

10.4.5 Initialization

There are two steps that must be performed prior to using the comparator block:

1. Enable the Band Gap inside the ADC by writing a 1 to the ADCBGPWD bit inside ADCCTL1.
2. Enable the comparator block by writing a 1 to the COMPDACEN bit in the COMPCTL register.

10.4.6 Digital Domain Manipulation

At the output of the comparator there are two more functional blocks that can be used to influence the behavior of the comparator output. They are:

1. Inverter circuit: Controlled by the CMPINV bit in the COMPCTL register; will apply a logical NOT to the output of the comparator. This function is asynchronous, while its control requires a clock present in order to change its value.
2. Qualification block: Controlled by the QUALSEL bit field in the COMPCTL register, and gated by the SYNCSEL bit in the COMPCTL register. This block can be used as a simple filter to only pass the output of the comparator once it is synchronized to the system clock. and qualified by the number of system clocks defined in QUALSEL bit field.

NOTE: Some downstream modules (such as the ePWM DC Submodule) may require a minimum comparator output pulse-width for correct operation.

Copyright © 2012–2019, Texas Instruments Incorporated

10.4.7 Comparator Registers

These devices have six comparators. Table 10-36 lists the registers for these modules.

Name	Address Range	Size(x16)	Description
COMP1	6400h – 641Fh	1	Comparator 1
COMP2	6420h – 643Fh	1	Comparator 2
COMP3	6440h - 645Fh	1	Comparator 3
COMP4	6460h - 647Fh	1	Comparator 4
COMP5	6480h - 649Fh	1	Comparator 5
COMP6	64A0h - 64BFh	1	Comparator 6

Table 10-36. Comparator Module Registers

Name	Address Range(base)	Size(x16)	Description
COMPCTL	0x0000 0000	1	Comparator Control[1]
Reserved	0x0000 0001	1	Reserved
COMPSTS	0x0000 0002	1	Comparator Output Status
Reserved	0x0000 0003	1	Reserved
Reserved	0x0000 0004	1	Reserved
Reserved	0x0000 0005	1	Reserved
DACVAL	0x0000 0006	1	10-bit DAC Value
Reserved	0x0000 0007 0x0000 0010	10	Reserved
DACTEST	0x0000 0011	1	DAC Test Register[2]
Reserved	0x0000 0012 0x0000 001F	14	Reserved

(1) This register is EALLOW protected.
(2) The DACTEST register is only available for COMP3/DAC3 and COMP6/DAC6.

10.4.7.1 Comparator Control (COMPCTL) Register

Figure 10-55. Comparator Control (COMPCTL) Register

15	9	8
Reserved		SYNCSEL
R-0		R/W-0

7	3	2	1	0
QUALSEL		CMPINV	COMPSOURCE	COMPDACEN
R/W-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-37. COMPCTL Register Field Descriptions

Bit	Field	Value	Description
15-9	Reserved		Reserved
8	SYNCSEL		Synchronization select for output of the comparator before being passed to ETPWM/GPIO blocks
		0	Asynchronous version of Comparator output is passed
		1	Synchronous version of comparator output is passed

Copyright © 2012–2019, Texas Instruments Incorporated

Table 10-37. COMPCTL Register Field Descriptions (continued)

Bit	Field	Value	Description
7-3	QUALSEL		Qualification Period for synchronized output of the comparator
		0h	Synchronized value of comparator is passed through
		1h	Input to the block must be consistent for 2 consecutive clocks before output of Qual block can change
		2h	Input to the block must be consistent for 3 consecutive clocks before output of Qual block can change
		...	...
		Fh	Input to the block must be consistent for 16 consecutive clocks before output of Qual block can change
2	CMPINV		Invert select for Comparator
		0	Output of comparator is passed
		1	Inverted output of comparator is passed
1	COMPSOURCE		Source select for comparator inverting input
		0	Inverting input of comparator connected to internal DAC
		1	Inverting input connected to external pin
0	COMPDACEN		Comparator/DAC Enable
		0	Comparator/DAC logic is powered down.
		1	Comparator/DAC logic is powered up.

10.4.7.2 Compare Output Status (COMPSTS) Register

Figure 10-56. Compare Output Status (COMPSTS) Register

15	1	0
Reserved		COMPSTS
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-38. Compare Output Status (COMPSTS) Register Field Descriptions

Bit	Field	Value	Description
15-1	Reserved		Reserved
0	COMPSTS		Logical latched value of the comparator

10.4.7.3 DAC Value (DACVAL) Register

Figure 10-57. DAC Value (DACVAL) Register

15	10	9	0
Reserved		DACVAL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10-39. DAC Value (DACVAL) Register Field Descriptions

Bit	Field	Value	Description
15-10	Reserved		Reserved
9-0	DACVAL	0-3FFh	DAC Value bits, scales the output of the DAC from 0 – 1023.

Copyright © 2012–2019, Texas Instruments Incorporated

10.4.7.4 DAC Test (DACTEST) Register

Figure 10-58. DAC Test (DACTEST) Register

15	1	0
Reserved		DACTEST
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 10-40. DAC Test (DACTEST) Register Field Descriptions

Bit	Field	Value	Description
15-1	Reserved		Reserved
0	DACTEST		DAC test
		0	DAC 3 is not connected to ADC1 or DAC6 is not connected to ADC2
		1	DAC 3 is connected to ADC1 or DAC6 is connected to ADC2
			Note: These bits are purely for test purposes only. Do not use COMP3 or COMP6 while these bits are set.

10.5 Analog Subsystem Software

This section describes all of the C/C++ Analog Subsystem functions included in controlSUITE™ and how to use them.

These functions are used to control the clocking for the Analog Subsystem.

10.5.1 C28 Analog Subsystem Functions

Four functions are defined in the F28M35x_GlobalPrototypes.h file:

- unsigned short InitAnalogSystemClock (unsigned short ClockDivider)
- unsigned short AnalogClockEnable (unsigned short AnalogConfigReg, unsigned short AnalogClockMask)
- unsigned short AnalogClockDisable (unsigned short AnalogConfigReg, unsigned short AnalogClockMask)
- unsigned short ReadAnalogClockStatus (unsigned short AnalogConfigReg)

There are also #define values defined in the F28M35x_AnalogSysCtrl_defines.h file which can be passed into the functions above.

10.5.1.1 InitAnalogSystemClock Function

The InitAnalogSystemClock function should be the first function called if the Analog Subsystem is to be used. This function allows configuration of the Analog Subsystem clock divider and checks if the Analog Common Interface Bus (ACIB) clock is enabled. The Analog Subsystem is clocked off of PLLSYSCLK.

Valid #define values for the ClockDivider parameter are located in the F28M35x_AnalogSysCtrl_defines.h file and listed below.

```
#define ACLKDIVOFF   0    // Clock divider off
#define ACLKDIV1     1    // Clock divider /1 (PLLSYSCLK/1)
#define ACLKDIV2     2    // Clock divider /2 (PLLSYSCLK/2)
#define ACLKDIV4     3    // Clock divider /4 (PLLSYSCLK/4)
#define ACLKDIV8     4    // Clock divider /8 (PLLSYSCLK/8)
```

NOTE: The maximum clock frequency for the Analog Subsystem clock is 37.5 MHz.

This function is successful if it returns a value of 0xA005.

Copyright © 2012–2019, Texas Instruments Incorporated

10.5.1.2 AnalogClockEnable Function

The AnalogClockEnable function enables the clock to each Analog Subsystem peripheral. This function requires two input parameters.

Valid #define values for the AnalogConfigReg parameter are located in the F28M35x_AnalogSysCtrl_defines.h file.

```
AnalogConfig1   1   Choose Analog Config Register 1 (used to enable ADC1)
AnalogConfig2   2   Choose Analog Config Register 2 (used to enable ADC2, COMP1,2,3,4,5,6)
```

Valid #define values for the AnalogClockMask parameter are located in the F28M35x_AnalogSysCtrl_defines.h file.

```
ADC1_ENABLE        0x0008   Mask to Enable ADC1
ADC2_ENABLE        0x8000   Mask to Enable ADC2
COMP1_ENABLE       0x0001   Mask to Enable COMP1
COMP2_ENABLE       0x0002   Mask to Enable COMP2
COMP3_ENABLE       0x0004   Mask to Enable COMP3
COMP4_ENABLE       0x0008   Mask to Enable COMP4
COMP5_ENABLE       0x0010   Mask to Enable COMP5
COMP6_ENABLE       0x0020   Mask to Enable COMP6
ANALOGCONFIG2ALL   0x803F   Mask to Enable ADC2, COMP1,2,3,4,5,6
```

NOTE: Multiple parameters can be used by ORing them together in the function call.

This function will return 0xFFFF while trying to enable the clocks, and 0 when the function completes successfully. This function must be run in a while loop. See example software in Section 10.5.2.

10.5.1.3 AnalogClockDisable Function

The AnalogClockDisable function disables the clock to each Analog Subsystem peripheral. This function requires two input parameters.

Valid #define values for the AnalogConfigReg parameter are located in the F28M35x_AnalogSysCtrl_defines.h file.

```
AnalogConfig1   1   Choose Analog Config Register 1 (used to enable ADC1)
AnalogConfig2   2   Choose Analog Config Register 2 (used to enable ADC2, COMP1,2,3,4,5,6)
```

Valid #define values for the AnalogClockMask parameter are located in the F28M35x_AnalogSysCtrl_defines.h file.

```
ADC1_ENABLE        0x0008   Mask to Enable ADC1
ADC2_ENABLE        0x8000   Mask to Enable ADC2
COMP1_ENABLE       0x0001   Mask to Enable COMP1
COMP2_ENABLE       0x0002   Mask to Enable COMP2
COMP3_ENABLE       0x0004   Mask to Enable COMP3
COMP4_ENABLE       0x0008   Mask to Enable COMP4
COMP5_ENABLE       0x0010   Mask to Enable COMP5
COMP6_ENABLE       0x0020   Mask to Enable COMP6
ANALOGCONFIG2ALL   0x803F   Mask to Enable ADC2, COMP1,2,3,4,5,6
```

NOTE: Multiple parameters can be used by ORing them together in the function call.

This function will return 0xFFFF while trying to disable the clocks, and 0 when the function completes successfully. This function must be run in a while loop. See example software in Section 10.5.2.

10.5.1.4 ReadAnalogClockStatus

The ReadAnalogClockStatus function is used to read the status of the Analog Subsystem Configuration 1 and 2 registers. These are the registers written to using the AnalogClockEnable and AnalogClockDisable functions.

Valid #define values for the AnalogConfigReg parameter are located in the F28M35x_AnalogSysCtrl_defines.h file.

Copyright © 2012–2019, Texas Instruments Incorporated

```
AnalogConfig1   1   Choose Analog Config Register 1 (used to read ADC1 clock status)
AnalogConfig2   2   Choose Analog Config Register 2 (used to read ADC2, COMP1,2,3,4,5,6 clock
status)
```

This function will return the status of the Analog Config Register 1 or 2 depending on the function parameter used.

10.5.2 C28 Analog Subsystem Software Example

Below is a software example showing how to use all the C28 Analog Subsystem functions in an application. The code initializes the Analog Subsystem and clock divider. It also enables ADC1, ADC2, COMP1, and COMP4 analog peripherals. Lastly, the example reads the status of the Analog Subsystem Config Registers to check for correctness.

```
unsigned short analoginit = 0;
unsigned short analogclock1 = 0;
unsigned short analogclock2 = 0;

analoginit = (**InitAnalogSystemClock)(ACLKDIV4);
// Initialize the Analog Subsystem and set the clock divider to divide by 4

EALLOW;
while((**AnalogClockEnable)(AnalogConfig1,ADC1_ENABLE));
// Enable ADC 1
while((**AnalogClockEnable)(AnalogConfig2,ADC2_ENABLE|COMP1_ENABLE|COMP4_ENABLE));
// Enable ADC 2, COMP1, and COMP4
analogclock1 = (**ReadAnalogClockStatus)(AnalogConfig1);
// Check Analog Config 1 Register for correct values (bit 3 should be set)
analogclock2 = (**ReadAnalogClockStatus)(AnalogConfig2);
// Check Analog Config 2 Register for correct values (bits 15,3,0 should be set)
EDIS;
```

NOTE: Bits in the Analog Config 1 and 2 registers that are not defined by the AnalogClockMask #define values are reserved.

NOTE: The ReadAnalogClockStatus function must be used to check the correctness of the Analog Config 1 and Analog Config 2 Registers if the header file functions (InitSysCtrl, InitAdc1, and InitAdc2) are not used to enable ADC1 or ADC2.

10.5.3 M3 Analog Subsystem Functions

Three functions are defined in the sysctl.h file:

- unsigned short AnalogClockEnable(unsigned short AnalogConfigReg, unsigned short AnalogClockMask)
- unsigned short AnalogClockDisable(unsigned short AnalogConfigReg, unsigned short AnalogClockMask)
- unsigned short ReadAnalogClockStatus(unsigned short AnalogConfigReg)

There are also #define values defined in the sysctl.h file which can be passed into the functions above.

10.5.3.1 AnalogClockEnable Function

The AnalogClockEnable function enables the clock to each Analog Subsystem peripheral. This function requires two input parameters.

Valid #define values for the AnalogConfigReg parameter are located in the sysctl.h file.

```
AnalogConfig1   1   Choose Analog Config Register 1 (used to enable ADC1)
AnalogConfig2   2   Choose Analog Config Register 2 (used to enable ADC2, COMP1,2,3,4,5,6)
```

Valid #define values for the AnalogClockMask parameter are located in the sysctl.h file.

Copyright © 2012–2019, Texas Instruments Incorporated

```
ADC1_ENABLE          0x0008   Mask to Enable ADC1
ADC2_ENABLE          0x8000   Mask to Enable ADC2
COMP1_ENABLE         0x0001   Mask to Enable COMP1
COMP2_ENABLE         0x0002   Mask to Enable COMP2
COMP3_ENABLE         0x0004   Mask to Enable COMP3
COMP4_ENABLE         0x0008   Mask to Enable COMP4
COMP5_ENABLE         0x0010   Mask to Enable COMP5
COMP6_ENABLE         0x0020   Mask to Enable COMP6
ANALOGCONFIG2ALL     0x803F   Mask to Enable ADC2, COMP1,2,3,4,5,6
```

NOTE: Multiple parameters can be used by ORing them together in the function call.

This function will return 0xFFFF while trying to enable the clocks, and 0 when the function completes successfully. This function must be run in a while loop. See example software in Section 10.5.4.

10.5.3.2 AnalogClockDisable Function

The AnalogClockDisable function disables the clock to each Analog Subsystem peripheral. This function requires two input parameters.

Valid #define values for the AnalogConfigReg parameter are located in the sysctl.h file.

```
AnalogConfig1    1   Choose Analog Config Register 1 (used to enable ADC1)
AnalogConfig2    2   Choose Analog Config Register 2 (used to enable ADC2, COMP1,2,3,4,5,6)
```

Valid #define values for the AnalogClockMask parameter are located in the sysctl.h file

```
ADC1_ENABLE          0x0008   Mask to Enable ADC1
ADC2_ENABLE          0x8000   Mask to Enable ADC2
COMP1_ENABLE         0x0001   Mask to Enable COMP1
COMP2_ENABLE         0x0002   Mask to Enable COMP2
COMP3_ENABLE         0x0004   Mask to Enable COMP3
COMP4_ENABLE         0x0008   Mask to Enable COMP4
COMP5_ENABLE         0x0010   Mask to Enable COMP5
COMP6_ENABLE         0x0020   Mask to Enable COMP6
ANALOGCONFIG2ALL     0x803F   Mask to Enable ADC2, COMP1,2,3,4,5,6
```

NOTE: Multiple parameters can be used by ORing them together in the function call.

This function will return 0xFFFF while trying to disable the clocks, and 0 when the function completes successfully. This function must be run in a while loop. See example software in Section 10.5.4.

10.5.3.3 ReadAnalogClockStatus

The ReadAnalogClockStatus function is used to read the status of the Analog Subsystem Configuration 1 and 2 registers. These are the registers written to using the AnalogClockEnable and AnalogClockDisable functions.

Valid #define values for the AnalogConfigReg parameter are located in the sysctl.h file.

```
AnalogConfig1    1   Choose Analog Config Register 1 (used to read ADC1 clock status)
AnalogConfig2    2   Choose Analog Config Register 2 (used to read ADC2, COMP1,2,3,4,5,6 clock
status)
```

This function will return the status of the Analog Config Register 1 or 2 depending on the function parameter used.

10.5.4 M3 Analog Subsystem Software Example

Below is a software example of how to use all the M3 Analog Subsystem functions in an application. The code enables ADC1, ADC2, COMP1, and COMP4 analog peripherals. The example also reads the status of the Analog Subsystem Config Registers to check for correctness.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: For the above example to work, the C28 Analog Subsystem function, InitAnalogSystemClock, must be called first.

```
unsigned short analogclock1 = 0;
unsigned short analogclock2 = 0;

HWREG  (SYSCTL_MWRALLOW) = 0xA5A5A5A5;
       // ALLOW reads and writes to protected registers;
while((**AnalogClockEnable)(AnalogConfig1,ADC1_ENABLE));
       // Enable ADC 1
while((**AnalogClockEnable)(AnalogConfig2,ADC2_ENABLE|COMP1_ENABLE|COMP4_ENABLE));
       // Enable ADC 2, COMP1, and COMP4
analogclock1 = (**ReadAnalogClockStatus)(AnalogConfig1);
       // Check Analog Config 1 Register for correct values (bit 3 should be set)
analogclock2 = (**ReadAnalogClockStatus)(AnalogConfig2);
       // Check Analog Config 2 Register for correct values (bits 15,3,0 should be set)
```

NOTE: Bits in the Analog Config 1 and 2 registers that are not defined by the AnalogClockMask #define values are reserved.

The functions above perform the same functionality as the C28 Analog Subsystem functions. These functions are available for software ease-of-use.

NOTE: The ReadAnalogClockStatus function must be used to check the correctness of the Analog Config 1 and Analog Config 2 Registers if the header file functions (InitSysCtrl, InitAdc1, and InitAdc2) are not used to enable ADC1 or ADC2.

Copyright © 2012–2019, Texas Instruments Incorporated

C28 Direct Memory Access (DMA) Module

The direct memory access (DMA) module provides a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. Additionally, the DMA has the capability to orthogonally rearrange the data as it is transferred as well as “ping-pong” data between buffers. These features are useful for structuring data into blocks for optimal CPU processing.

Topic **Page**

11.1 Introduction **912**
11.2 Architecture **912**
11.3 Pipeline Timing and Throughput **916**
11.4 CPU Arbitration **917**
11.5 Channel Priority **918**
11.6 Address Pointer and Transfer Control **920**
11.7 Overrun Detection Feature **924**
11.8 Register Descriptions **926**

Copyright © 2012–2019, Texas Instruments Incorporated

11.1 Introduction

The strength of a controller is not measured purely in processor speed, but in total system capabilities. As a part of the equation, any time the CPU bandwidth for a given function can be reduced, the greater the system capabilities. Many times applications spend a significant amount of their bandwidth moving data, whether it is from off-chip memory to on-chip memory, or from a peripheral such as an analog-to-digital converter (ADC) to RAM, or even from one peripheral to another. Furthermore, many times this data comes in a format that is not conducive to the optimum processing powers of the CPU. The DMA module described in this reference guide has the ability to free up CPU bandwidth and rearrange the data into a pattern for more streamlined processing.

The DMA module is an event-based machine, meaning it requires a peripheral interrupt trigger to start a DMA transfer. Although it can be made into a periodic time-driven machine by configuring a timer as the interrupt trigger source, there is no mechanism within the module itself to start memory transfers periodically. The interrupt trigger source for each of the six DMA channels can be configured separately and each channel contains its own independent PIE interrupt to let the CPU know when a DMA transfers has either started or completed. Five of the six channels are exactly the same, while Channel 1 has one additional feature: the ability to be configured at a higher priority than the others. At the heart of the DMA is a state machine and tightly coupled address control logic. It is this address control logic that allows for rearrangement of the block of data during the transfer as well as the process of ping-ponging data between buffers. Each of these features, along with others, will be discussed in detail in this document.

DMA Overview:

- 6 channels with independent PIE interrupts
- Peripheral interrupt trigger sources
  - ADC interrupts 1-4
  - Multichannel buffered serial port transmit and receive
  - XINT1-3
  - CPU Timers 0-2
  - ePWM1-9 ADCSOCA and ADSOCB signals
  - Software
- Data sources/destinations:
  - L5-L8 32K x 16 SARAM
  - ADC memory bus mapped result registers
  - McBSP transmit and receive buffers
  - ePWM1-8 / HRPWM1-8
- Word Size: 16-bit or 32-bit (McBSP limited to 16-bit)
- Throughput: 4 cycles/word (5 cycles/word for McBSP reads)

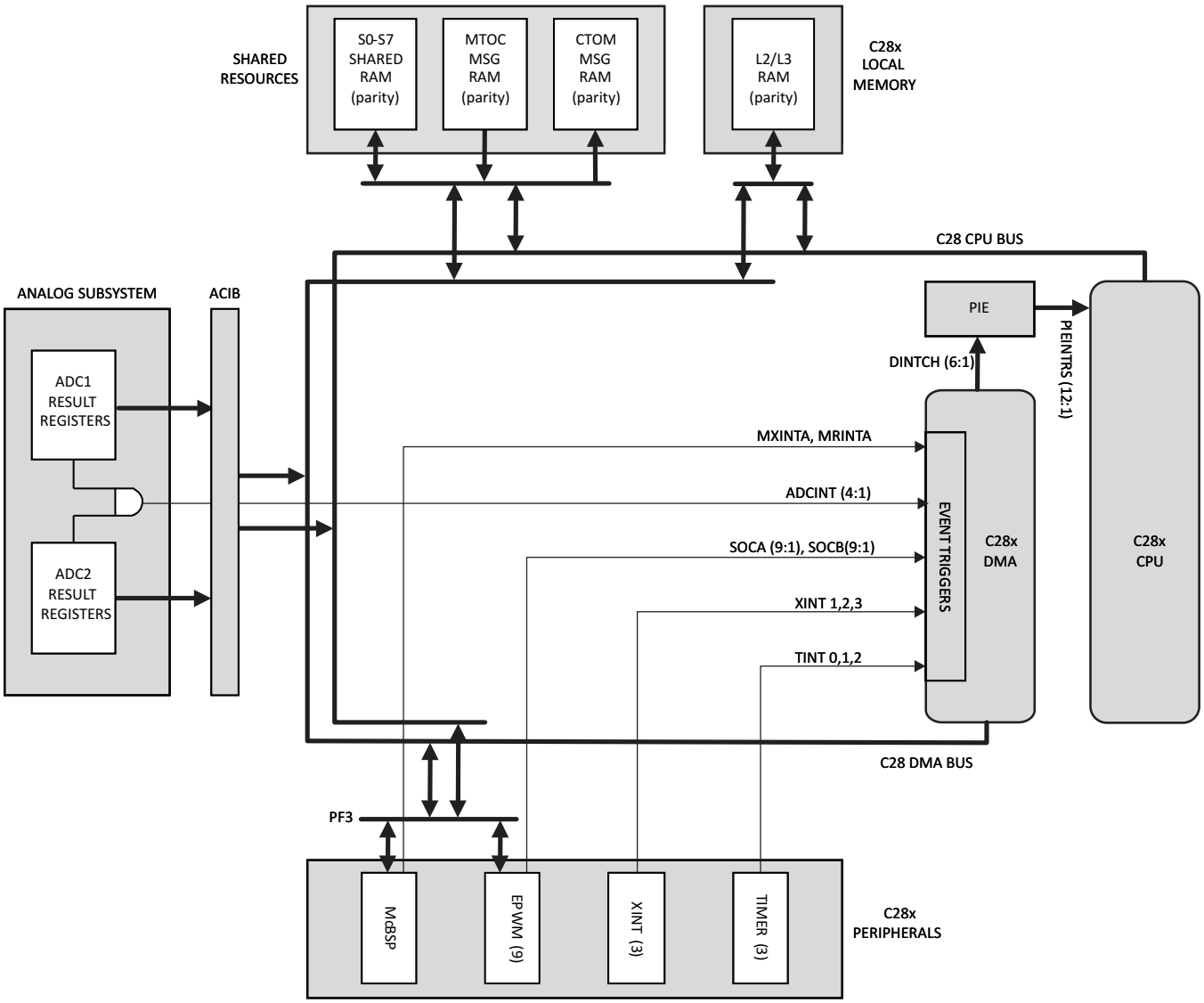
11.2 Architecture

11.2.1 Block Diagram

Figure 11-1 shows a device-level block diagram of the DMA.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 11-1. DMA Block Diagram



11.2.2 Peripheral Interrupt Event Trigger Sources

The peripheral interrupt event trigger can be independently configured as one twenty-nine different sources for each of the six DMA channels. Included in these sources are three external interrupt signals which can be connected to most of the general-purpose input/output (GPIO) pins on the device. This adds significant flexibility to the event trigger capabilities. A bit field called PERINTSEL in the MODE register of each channel is used to select that channels interrupt trigger source. An active peripheral interrupt trigger will be latched into the PERINTFLG bit of the CONTROL register, and if the respective interrupt and DMA channel is enabled (see the MODE.CHx[PERINTE] and CONTROL.CHx[RUNSTS] bits), it will be serviced by the DMA channel. Upon receipt of a peripheral interrupt event signal, the DMA will automatically send a clear signal to the interrupt source so that subsequent interrupt events will occur.

Regardless of the value of the MODE.CHx[PERINTSEL] bit field, software can always force a trigger by using the CONTROL.CHx[PERINTFRC] bit. Likewise, software can always clear a pending DMA trigger using the CONTROL.CHx[PERINTCLR] bit.

Copyright © 2012–2019, Texas Instruments Incorporated

Once a particular interrupt trigger sets a channel's PERINTFLG bit, the bit stays pending until the priority logic of the state machine starts the burst transfer for that channel. Once the burst transfer starts, the flag is cleared. If a new interrupt trigger is generated while a burst is in progress, the burst will complete before responding to the new interrupt trigger (after proper prioritization). If a third interrupt trigger occurs before the pending interrupt is serviced, an error flag is set in the CONTROL.CHx[OVRFLG] bit. If a peripheral interrupt trigger occurs at the same time as the latched flag is being cleared, the peripheral interrupt trigger has priority and the PERINTFLG will remain set.

Figure 11-2 shows a diagram of the trigger select circuit. See the MODE.CHx[PERINTSEL] bit field description for the complete list of peripheral interrupt trigger sources.

Figure 11-2. Peripheral Interrupt Trigger Input Diagram

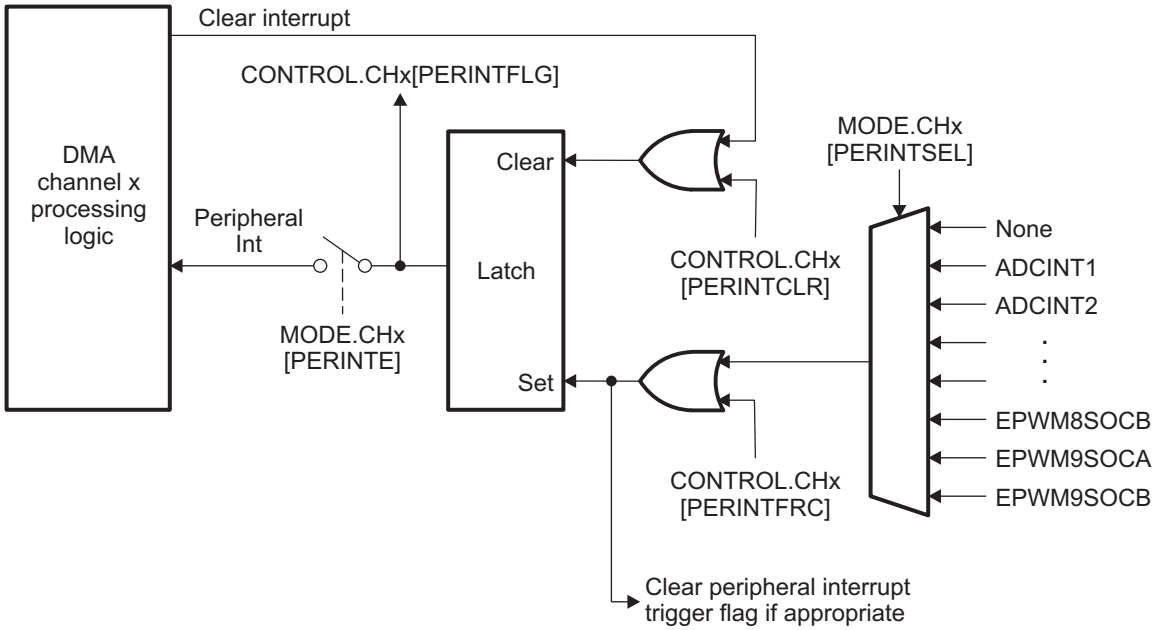


Table 11-1 shows the interrupt trigger source options that are available for each channel.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 11-1. Peripheral Interrupt Trigger Source Options

Peripheral	Interrupt Trigger Source
CPU	DMA Software bit (CHx.CONTROL.PERINTFRC) only
ADC	ADCINT 1
	ADCINT 2
	ADCINT3
	ADCINT4
External Interrupts	External Interrupt 1
	External Interrupt 2
	External Interrupt 3
CPU Timers	Timer 0 Overflow
	Timer 1 Overflow
	Timer 2 Overflow
McBSP	McBSP Transmit Buffer Empty
	McBSP Receive Buffer Full
ePWM1	ADC Start of Conversion A
	ADC Start of Conversion B
ePWM2	ADC Start of Conversion A
	ADC Start of Conversion B
ePWM3	ADC Start of Conversion A
	ADC Start of Conversion B
ePWM4	ADC Start of Conversion A
	ADC Start of Conversion B
ePWM5	ADC Start of Conversion A
	ADC Start of Conversion B
ePWM6	ADC Start of Conversion A
	ADC Start of Conversion B
ePWM7	ADC Start of Conversion A
	ADC Start of Conversion B
ePWM8	ADC Start of Conversion A
	ADC Start of Conversion B
ePWM9	ADC Start of Conversion A
	ADC Start of Conversion B

11.2.3 DMA Bus

The DMA bus architecture consists of a 22-bit address bus, a 32-bit data read bus, and a 32-bit data write bus. Memories and register locations connected to the DMA bus are via interfaces that sometimes share resources with the CPU memory or peripheral bus. Arbitration rules are defined in Section 11.4. The following resources are connected to the DMA bus:

- L2/L3 C28x local RAM
- CTOM MSG RAM
- MTOC MSG RAM
- S0-S7 shared RAM
- ADC 1 memory-mapped result registers
- ADC 2 memory-mapped result registers
- McBSP data receive registers (DRR2/DRR1) and data transmit registers (DXR2/DXR1)
- ePWM1-9/HRPWM1-8 registers

Copyright © 2012–2019, Texas Instruments Incorporated

11.3 Pipeline Timing and Throughput

The DMA consists of a 4-stage pipeline as shown in Figure 11-3. The one exception to this is when a DMA channel is configured to have the McBSP as its data source. A read of a McBSP DRR register stalls the DMA bus for one cycle during the read portion of the transfer, as shown in Figure 11-4.

Figure 11-3. 4-Stage Pipeline DMA Transfer

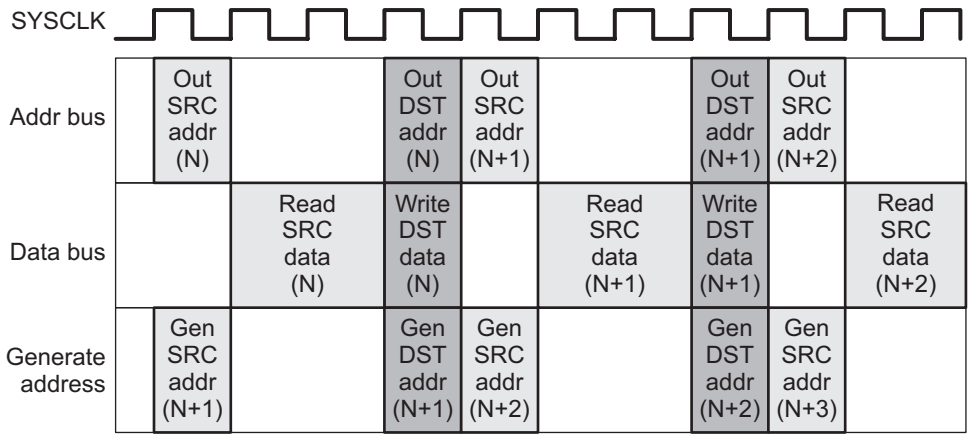
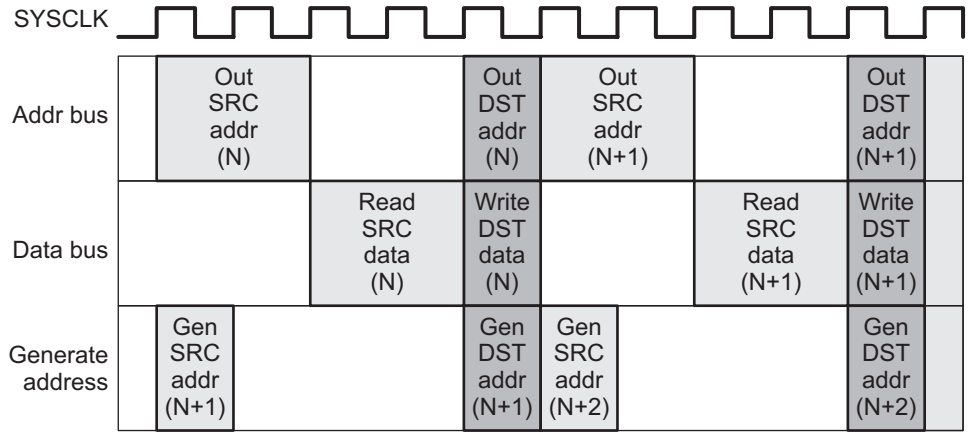


Figure 11-4. 4-Stage Pipeline With One Read Stall (McBSP as source)



In addition to the pipeline there are a few other behaviors of the DMA that affect its total throughput:

- A 1-cycle delay is added at the beginning of each burst
- A 1-cycle delay is added when returning from a CH1 high priority interrupt
- Collisions with the CPU may add delay slots (see Section 11.4)
- Arbitration inside the analog common interface bus (ACIB) may add delay slots when reading ADC result registers
- 32-bit transfers run at double the speed of a 16-bit transfer (it takes the same amount of time to transfer a 32-bit word as it does a 16-bit word)

For example, to transfer 128 16-bit words from L2 RAM to L3 RAM a channel can be configured to transfer 8 bursts of 16 words/burst. This will give:

8 bursts * [(4 cycles/word * 16 words/burst) + 1] = 520 cycles

If instead the channel were configured to transfer the same amount of data 32 bits at a time (the word size is configured to 32 bits) the transfer would take:

Copyright © 2012–2019, Texas Instruments Incorporated

8 bursts * [(4 cycles/word * 8 words/burst) + 1] = 264 cycles

11.3.1 DMA Read Access of ADC Registers

DMA access to the ADC result registers is achieved via the ACIB. The ACIB path produces different cycle counts for DMA transfers when compared to simple transfers such as RAM-to-RAM data transfers. Additional cycles are needed for the "Read SRC data (N)" portion of the 4-Stage Pipeline DMA Transfer timing diagram.

For improved efficiency, the ACIB automatically uses a 64-bit read mode when the ADC result registers are accessed with burst reads and the following two conditions are met:

1. The DMA is set to 32-bit transfer mode, and
2. The DMA start address is aligned to a 64-bit boundary (Note: The start address of ADC1 and ADC2 result registers fall on 64-bit boundaries)

The total number of Control Subsystem clock cycles required for transferring ADC results using the DMA in conjunction with the 64-bit ACIB read mode can be generalized as:

{ [ACIB Read Cycles] + [DMA Burst Cycles] } * Number-of-DMA-Bursts

ACIB Read Cycles is calculated as:

16 ACIB Cycles * (Bits per DMA Burst / 64 bits) * (Control Subsys Freq / Analog Subsys Freq)

DMA Burst Cycles is calculated as:

2 Cycles * (Bits per DMA Burst / 32 bits)

For example, to transfer 16 16-bit words (16 ADC Results) to RAM, the DMA burst size is set to 16 words so that the total 256-bit transfer is completed with one DMA burst. The cycles required for this transfer is:

{ [16 * (256 / 64) * (150 / 37.5)] + [2 * (256 / 32)] } * 1 = 272 Cycles

11.4 CPU Arbitration

Typically, DMA activity is independent of the CPU activity. Under the circumstance where both the DMA and the CPU are attempting to access memory or a peripheral register within the same interface concurrently, an arbitration procedure will occur. Any combined accesses between the different interfaces, or where the CPU access is outside of the interface that the DMA is accessing do not create a conflict. For example, if the CPU is accessing ePWM while the DMA is simultaneously accessing McBSP, it will create a conflict because both ePWM and McBSP reside in a common interface (peripheral frame 3). However, if the CPU is accessing shared RAM while the DMA is accessing message RAM, there will be no conflict, since these two memories are located in different interfaces (shared resources and C28x local memory).

The interfaces which internally contain conflicts are:

- L2/L3 C28x local RAM
- CTOM MSG RAM
- MTOC MSG RAM
- S0-S7 shared RAM
- McBSP peripheral frame 3
- ePWM/HRPWM peripheral frame 3

If the CPU and the DMA make an access to the same RAM block in the same cycle, the conflict is resolved with round-priority scheme. If the CPU and the DMA make an access to the same peripheral frame in the same cycle, the DMA has priority and the CPU is stalled.

If a CPU access to an interface is in progress and another CPU access to the same interface is pending, for example, the CPU is performing a write operation and a read operation from the CPU is pending, then a DMA access to that same interface has priority over the pending CPU access when the current CPU access completes.

Copyright © 2012–2019, Texas Instruments Incorporated

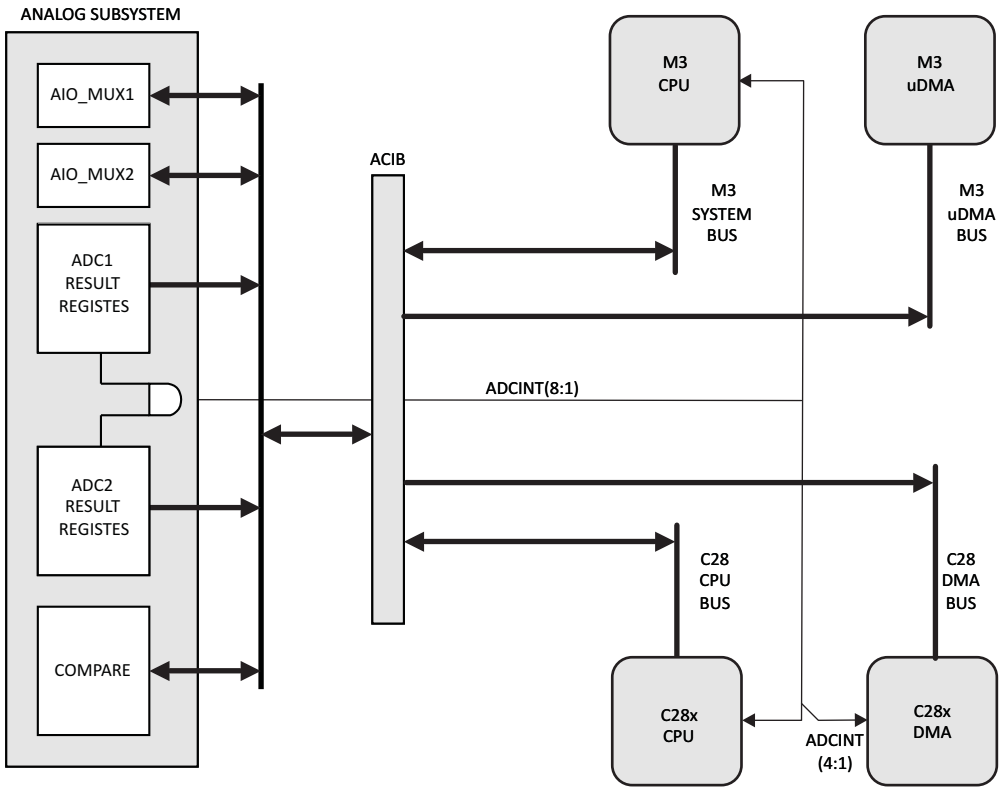
NOTE: If the CPU is performing a read-modify-write operation and the DMA performs a write to the same location, the DMA write may be lost if the operation occurs in between the CPU read and the CPU write. For this reason, it is advised not to mix such CPU accesses with DMA accesses to the same locations.

In the case of RAM, a ping-pong scheme can be implemented to avoid the CPU and the DMA accessing the same RAM block concurrently, thus avoiding any stalls or corruption issues.

11.4.1 Arbitration when Accessing the Analog Subsystem

The DMA read cycles to the analog subsystem must pass through the analog common interface bus (ACIB) when accessing ADC result registers. The analog subsystem can be accessed by four masters – C28 CPU, C28 DMA, M3 CPU, and M3 DMA. In a case where multiple masters are simultaneously attempting to access analog subsystem peripherals, an arbitration procedure will occur. The amount of time that DMA bus cycles may be delayed depends on how many of the other three masters are trying to access the analog subsystem at the same time. The ACIB bus arbiter uses the round-robin algorithm to determine which bus cycle gets through and which bus cycles have to wait at a given time. Figure 11-5 illustrates this process.

Figure 11-5. Arbitration when Accessing ACIB



11.5 Channel Priority

Two priority schemes exist when determining channel priority: Round-robin mode and Channel 1 high-priority mode.

11.5.1 Round-Robin Mode

In this mode, all channels have *equal* priority and each enabled channel is serviced in round-robin fashion as follows:

Copyright © 2012–2019, Texas Instruments Incorporated

CH1 → CH2 → CH3 → CH4 → CH5 → CH6 → CH1 → CH2 → …

In the case above, after each channel has transferred a burst of words, the next channel is serviced. You can specify the size of the burst for each channel. Once CH6 (or the last enabled channel) has been serviced, and no other channels are pending, the round-robin state machine enters an idle state.

From the idle state, channel 1 (if enabled) is always serviced first. However, if the DMA is currently processing another channel x, all other pending channels between x and the end of the round are serviced before CH1. It is in this sense that all the channels are of *equal* priority. For instance, take an example where CH1, CH4, and CH5 are enabled in round-robin mode and CH4 is currently being processed. Then CH1 and CH5 both receive an interrupt trigger from their respective peripherals before CH4 completes. CH1 and CH5 are now both pending. When CH4 completes its burst, CH5 will be serviced next. Only after CH5 completes will CH1 be serviced. Upon completion of CH1, if there are no more channels pending, the round-robin state machine will enter an idle state.

A more complicated example is shown below:

- Assume all channels are enabled, and the DMA is in an idle state,
- Initially a trigger occurs on CH1, CH3, and CH5 on the same cycle,
- When the CH1 burst transfer starts, requests from CH3 and CH5 are pending,
- Before completion of the CH1 burst, the DMA receives a request from CH2. Now the pending requests are from CH2, CH3, and CH5,
- After completing the CH1 burst, CH2 will be serviced since it is next in the round-robin scheme after CH1.
- After the burst from CH2 is finished, the CH3 burst will be serviced, followed by CH5 burst.
- Now while the CH5 burst is being serviced, the DMA receives a request from CH1, CH3, and CH6.
- The burst from CH6 will start after the completion of the CH5 burst since it is the next channel after CH5 in the round-robin scheme.
- This will be followed by the CH1 burst and then the CH3 burst
- After the CH3 burst finishes, assuming no more triggers have occurred, the round-robin state machine will enter an idle state.

The round-robin state machine may be reset to the idle state via the DMACTRL[PRIORITYRESET] bit.

11.5.2 Channel 1 High Priority Mode

In this mode, if a CH1 trigger occurs, the current word transfer or the current + 1 word transfer (depends on which phase of the current DMA transfer the new CH1 trigger occurred) on any other channel is completed (not the complete burst), execution is halted, and CH1 is serviced for the complete burst count. When the CH1 burst is complete, execution returns to the channel that was active when the CH1 trigger occurred. All other channels have equal priority and each enabled channel is serviced in round-robin fashion as follows:

Higher Priority:	CH1
Lower priority:	CH2 → CH3 → CH4 → CH5 → CH6 → CH2 → …

Given an example where CH1, CH4 and CH5 are enabled in Channel 1 High Priority Mode and CH4 is currently being processed. Then CH1 and CH5 both receive an interrupt trigger from their respective peripherals before CH4 completes. CH1 and CH5 are now both pending. When the current CH4 word transfer is completed, regardless of whether the DMA has completed the entire CH4 burst, CH4 execution will be suspended and CH1 will be serviced. After the CH1 burst completes, CH4 will resume execution.

Upon completion of CH4, CH5 will be serviced. After CH5 completes, if there are no more channels pending, the round-robin state machine will enter an idle state.

Typically Channel 1 would be used in this mode for the ADC, since its data rate is so high. However, Channel 1 High Priority Mode may be used in conjunction with any peripheral.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: High-priority mode and ONESHOT mode may not be used at the same time on channel 1. Other channels may use ONESHOT mode when channel 1 is in high-priority mode.

11.6 Address Pointer and Transfer Control

The DMA state machine is, at its most basic level, two nested loops. The inner loop transfers a burst of data when a peripheral interrupt trigger is received. A burst is the smallest amount of data that can be transferred at one time and its size is defined by the BURST_SIZE register for each channel. The BURST_SIZE register allows a maximum of 32 sixteen-bit words to be transferred in one burst. The outer loop, whose size is set by the TRANSFER_SIZE register for each channel, defines how many bursts are performed in the entire transfer. Since TRANSFER_SIZE is a 16-bit register, the total size of a transfer allowed is well beyond any practical requirement. One CPU interrupt is generated, if enabled, for each transfer. This interrupt can be configured to occur at the beginning or the end of the transfer via the MODE.CHx[CHINTMODE] bit.

In the default setting of the MODE.CHx[ONESHOT] bit, the DMA transfers one burst of data each time a peripheral interrupt trigger is received. After the burst is completed, the state machine moves on to the next pending channel in the priority scheme, even if another trigger for the channel just completed is pending. This feature keeps any single channel from monopolizing the DMA bus. If a transfer of more than the maximum number of words per burst is desired for a single trigger, the MODE.CHx[ONESHOT] bit can be set to complete the entire transfer when triggered. Care is advised when using this mode, since this can create a condition where one trigger uses up the majority of the DMA bandwidth.

Each DMA channel contains a shadowed address pointer for the source and the destination address. These pointers, SRC_ADDR and DST_ADDR, can be independently controlled during the state machine operation. At the beginning of each transfer, the shadowed version of each pointer is copied into its respective active register. During the burst loop, after each word is transferred, the signed value contained in the appropriate source or destination BURST_STEP register is added to the active SRC/DST_ADDR register. During the transfer loop, after each burst is complete, there are two methods that can be used to modify the active address pointer. The first, and default, method is by adding the signed value contained in the SRC/DST_TRANSFER_STEP register to the appropriate pointer. The second is via a process called wrapping, where a wrap address is loaded into the active address pointer. When a wrap procedure occurs, the associated SRC/DST_TRANSFER_STEP register has no effect.

Address wrapping occurs when a number of bursts specified by the appropriate SRC/DST_WRAP_SIZE register completes. Each DMA channel contains two shadowed wrap address pointers, SRC_BEG_ADDR and DST_BEG_ADDR, allowing the source and destination wrapping to be independent of each other. Like the SRC_ADDR and DST_ADDR registers, the active SRC/DST_BEG_ADDR registers are loaded from their shadow counterpart at the beginning of a transfer. When the specified number of bursts has occurred, a two part wrap procedure takes place:

- The appropriate active SRC/DST_BEG_ADDR register is incremented by the signed value contained in the SRC/DST_WRAP_STEP register, then
- The new active SRC/DST_BEG_ADDR register is loaded into the active SRC/DST_ADDR register.

Additionally the wrap counter (SRC/DST_WRAP_COUNT) register is reloaded with the SRC/DST_WRAP_SIZE value to setup the next wrap period. This allows the channel to wrap multiple times within a single transfer. Combined with the first bullet above, this allows the channel to address multiple buffers within a single transfer.

The DMA contains both an active and shadow set of the following address pointers. When a DMA transfer begins, the shadow register set is copied to the active working set of registers. This allows you to program the values of the shadow registers for the next transfer while the DMA works with the active set. It also allows you to implement Ping-Pong buffer schemes without disrupting the DMA channel execution.

Source/Destination Address Pointers (SRC/DST_ADDR)— The value written into the shadow register is the start address of the first location where data is read or written to.

At the beginning of a transfer the shadow register is copied into the active register. The active register performs as the current address pointer.

Source/Destination Begin Address Pointers (SRC/DST_BEG_ADDR)— This is the wrap pointer.

Copyright © 2012–2019, Texas Instruments Incorporated

The value written into the shadow register will be loaded into the active register at the start of a transfer. On a wrap condition, the active register will be incremented by the signed value in the appropriate SRC/DST_WRAP_STEP register prior to being loaded into the active SRC/DST_ADDR register.

For each channel, the transfer process can be controlled with the following size values:

Source and Destination Burst Size (BURST_SIZE): — This specifies the number of words to be transferred in a burst.

This value is loaded into the BURST_COUNT register at the beginning of each burst. The BURST_COUNT decrements each word that is transferred and when it reaches a zero value, the burst is complete, indicating that the next channel can be serviced. The behavior of the current channel is defined by the ONE_SHOT bit in the MODE register. The maximum size of the burst is dictated by the type of peripheral. For the ADC, the burst size could be all 16 registers (if all 16 registers are used). For a McBSP peripheral, the burst size is limited to 1 since there is no FIFO and the receive or transmit data register must be loaded or copied every word transferred. For RAM the burst size can be up to the maximum allowed by the BURST_SIZE register, which is 32.

Source and Destination Transfer Size (TRANSFER_SIZE): — This specifies the number of bursts to be transferred before per CPU interrupt (if enabled).

Whether this interrupt is generated at the beginning or the end of the transfer is defined in the CHINTMODE bit in the MODE register. Whether the channel remains enabled or not after the transfer is completed is defined by the CONTINUOUS bit in the MODE register. The TRANSFER_SIZE register is loaded into the TRANSFER_COUNT register at the beginning of each transfer. The TRANSFER_COUNT register keeps track of how many bursts of data the channel has transferred and when it reaches zero, the DMA transfer is complete.

Source/Destination Wrap Size (SRC/DST_WRAP_SIZE)— This specifies the number of bursts to be transferred before the current address pointer wraps around to the beginning.

This feature is used to implement a circular addressing type function. This value is loaded into the appropriate SRC/DST_WRAP_COUNT register at the beginning of each transfer. The SRC/DST_WRAP_COUNT registers keep track of how many bursts of data the channel has transferred and when they reaches zero, the wrap procedure is performed on the appropriate source or destination address pointer. A separate size and count register is allocated for source and destination pointers. To *disable* the wrap function, assign the value of these registers to be larger than the TRANSFER_SIZE.

NOTE: The value written to the SIZE registers is one less than the intended size. So, to transfer three 16-bit words, the value 2 should be placed in the SIZE register.

Regardless of the state of the DATASIZE bit, the value specified in the SIZE registers are for 16-bit addresses. So, to transfer three 32-bit words, the value 5 should be placed in the SIZE register.

For each source/destination pointer, the address changes can be controlled with the following step values:

Source/Destination Burst Step (SRC/DST_BURST_STEP)— Within each burst transfer, the address source and destination step sizes are specified by these registers.

This value is a signed 2's compliment number so that the address pointer can be incremented or decremented as required. If no increment is desired, such as when accessing the McBSP data receive or transmit registers, the value of these registers should be set to zero.

Source/Destination Transfer Step (SRC/DST_TRANSFER_STEP)— This specifies the address offset to start the next burst transfer after completing the current burst transfer.

This is used in cases where registers or data memory locations are spaced at constant intervals. This value is a signed 2's compliment number so that the address pointer can be incremented or decremented as required.

Copyright © 2012–2019, Texas Instruments Incorporated

Source/Destination Wrap Step (SRC/DST_WRAP_STEP): — When the wrap counter reaches zero, this value specifies the number of words to add/subtract from the BEG_ADDR pointer and hence sets the new start address.

This implements a circular type of addressing mode, useful in many applications. This value is a signed 2's compliment number so that the address pointer can be incremented or decremented as required.

NOTE: Regardless of the state of the DATASIZE bit, the value specified in the STEP registers are for 16-bit addresses. So, to increment one 32-bit address, a value of 2 should be placed in these registers.

Three modes are provided to control the way the state machine behaves during the burst loop and the transfer loop:

One Shot Mode (ONESHOT)— If one shot mode is enabled when an interrupt event trigger occurs, the DMA will continue transferring data in bursts until TRANSFER_COUNT is zero. If one shot mode is disabled, then an interrupt event trigger is required for each burst transfer and this will continue until TRANSFER_COUNT is zero.

NOTE: When ONESHOT mode is enabled, the DMA will continuously transfer bursts of data on the given channel until the TRANSFER_COUNT value is zero. This could potentially hog the bandwidth of a peripheral and cause long CPU stalls to occur. To avoid this, you could configure a CPU timer (or similar) and disable ONESHOT so as to avoid this situation.

High-priority mode and ONESHOT mode may not be used at the same time on channel 1. Other channels may use ONESHOT mode when channel 1 is in high-priority mode.

Continuous Mode (CONTINUOUS)— If continuous mode is disabled the RUNSTS bit in the CONTROL register is cleared at the end of the transfer, disabling the DMA channel.

The channel must be re-enabled by setting the RUN bit in the CONTROL register before another transfer can be started on that channel. If the continuous mode is enabled the RUNSTS bit is not cleared at the end of the transfer.

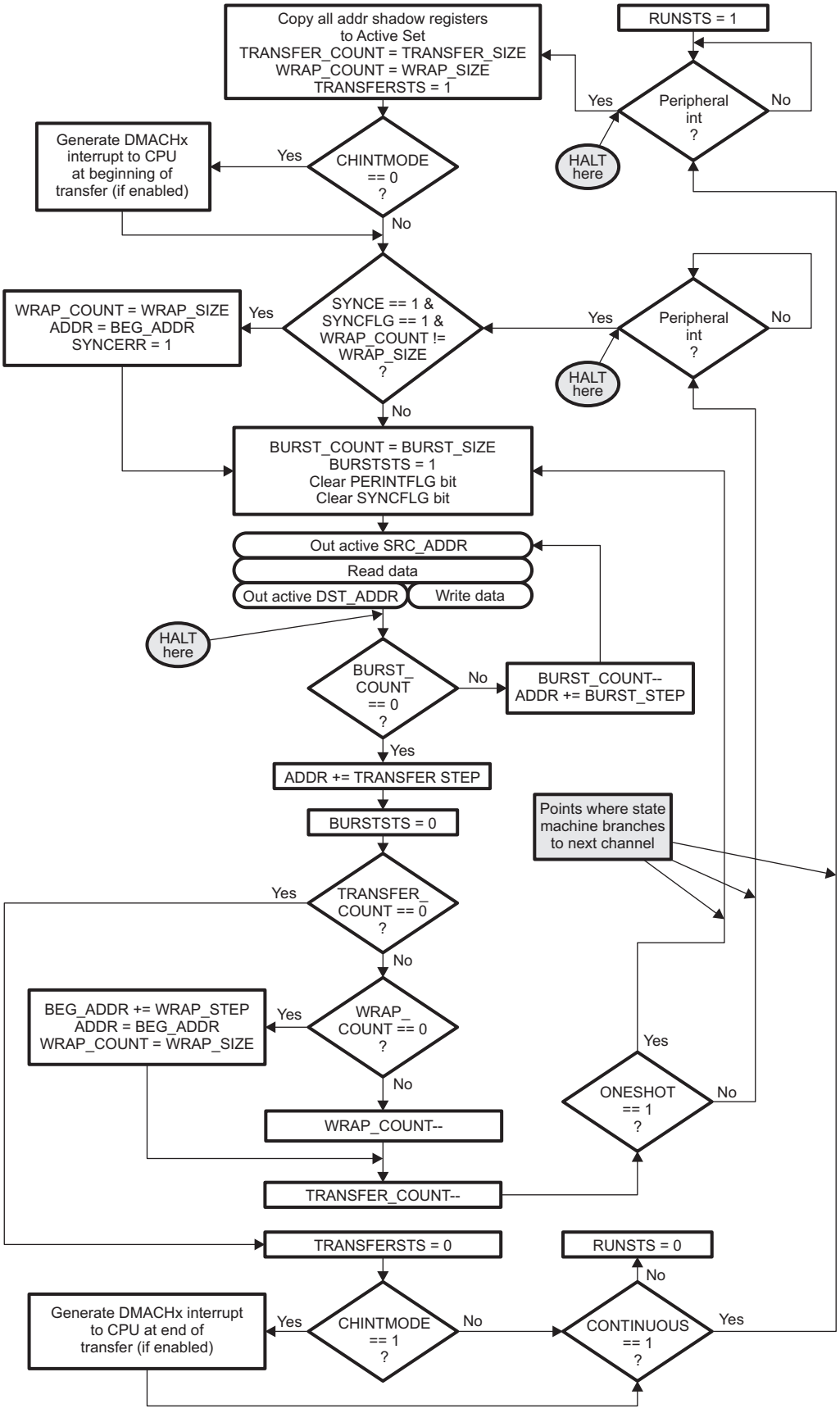
Channel Interrupt Mode (CHINTMODE)— This mode bit selects whether the DMA interrupt from the respective channel is generated at the beginning of a new transfer or at the end of the transfer.

If implementing a ping-pong buffer scheme with continuous mode of operation, then the interrupt would be generated at the beginning, just after the working registers are copied to the shadow set. If the DMA does not operate in continuous mode, then the interrupt is typically generated at the end when the transfer is complete.

All of the above features and modes are shown in Figure 11-6.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 11-6. DMA State Diagram



Copyright © 2012–2019, Texas Instruments Incorporated

The following items are in reference to Figure 11-6.

- The *HALT* points represent where the channel halts operation when interrupted by a high priority channel 1 trigger, or when the HALT command is set, or when an emulation halt is issued and the FREE bit is cleared to 0.
- The ADDR registers are not affected by BEG_ADDR at the start of a transfer. BEG_ADDR only affects the ADDR registers on a wrap or sync error. Following is what happens to each of the ADDR registers when a transfer first starts:
  - BEG_ADDR_SHADOW remains unchanged.
  - ADDR_SHADOW remains unchanged.
  - BEG_ADDR = BEG_ADDR_SHADOW
  - ADDR = ADDR_SHADOW
- The active registers get updated when a wrap occurs. The shadow registers remain unchanged. Specifically:
  - BEG_ADDR_SHADOW remains unchanged.
  - ADDR_SHADOW remains unchanged.
  - BEG_ADDR += WRAP_STEP
  - ADDR = BEG_ADDR
- The active registers get updated when a sync error occurs. The shadow registers remain unchanged. Specifically:
  - BEG_ADDR_SHADOW remains unchanged.
  - ADDR_SHADOW remains unchanged.
  - BEG_ADDR remains unchanged.
  - ADDR = BEG_ADDR

Probably the easiest way to remember all this is that:

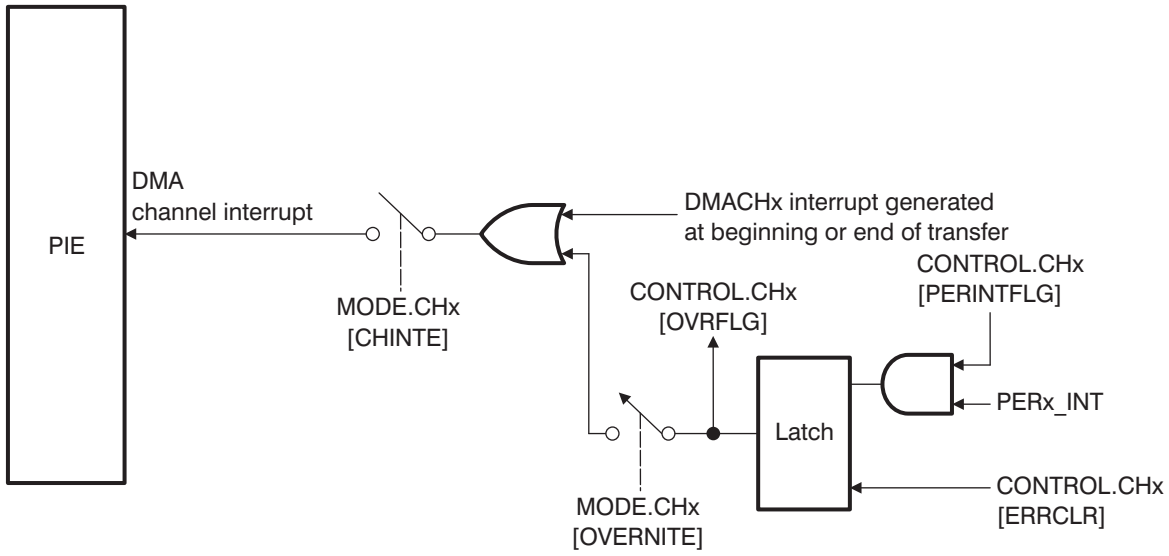
- The shadow registers never change except by software.
- The active registers never change except by hardware, and a shadow register is only copied into its own active register, never an active register by another name.

11.7 Overrun Detection Feature

The DMA contains overrun detection logic. When a peripheral event trigger is received by the DMA, the PERINTFLG bit in the CONTROL register is set, pending the channel to the DMA state machine. When the burst for that channel is started, the PERINTFLG is cleared. If however, between the time that the PERINTFLG bit is set by an event trigger and cleared by the start of the burst, an additional event trigger arrives, the second trigger will be lost. This condition will set the OVRFLG bit in the CONTROL register as in Figure 11-7. If the overrun interrupt is enabled then the channel interrupt will be generated to the PIE module.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 11-7. Overrun Detection Logic



Copyright © 2012–2019, Texas Instruments Incorporated

11.8 Register Descriptions

The complete DMA register set is shown in Table 11-2.

Table 11-2. DMA Register Summary[1]

Address	Acronym	Description	Section
DMA Control, Mode and Status Registers			
0x1000	DMACTRL	DMA Control Register	Section 11.8.1
0x1001	DEBUGCTRL	Debug Control Register	Section 11.8.2
0x1002	REVISION	Peripheral Revision Register	Section 11.8.3
0x1003	Reserved	Reserved	
0x1004	PRIORITYCTRL1	Priority Control Register 1	Section 11.8.4
0x1005	Reserved	Reserved	
0x1006	PRIORITYSTAT	Priority Status Register	Table 11-7
0x1007 0x101F	Reserved	Reserved	
DMA Channel 1 Registers			
0x1020	MODE	Mode Register	Section 11.8.6
0x1021	CONTROL	Control Register	Section 11.8.7
0x1022	BURST_SIZE	Burst Size Register	Section 11.8.8
0x1023	BURST_COUNT	Burst Count Register	Section 11.8.9
0x1024	SRC_BURST_STEP	Source Burst Step Size Register	Section 11.8.10
0x1025	DST_BURST_STEP	Destination Burst Step Size Register	Section 11.8.11
0x1026	TRANSFER_SIZE	Transfer Size Register	Table 11-13
0x1027	TRANSFER_COUNT	Transfer Count Register	Section 11.8.13
0x1028	SRC_TRANSFER_STEP	Source Transfer Step Size Register	Section 11.8.14
0x1029	DST_TRANSFER_STEP	Destination Transfer Step Size Register	Section 11.8.15
0x102A	SRC_WRAP_SIZE	Source Wrap Size Register	Section 11.8.16
0x102B	SRC_WRAP_COUNT	Source Wrap Count Register	Section 11.8.17
0x102C	SRC_WRAP_STEP	Source Wrap Step Size Register	Section 11.8.18
0x102D	DST_WRAP_SIZE	Destination Wrap Size Register	Section 11.8.16
0x102E	DST_WRAP_COUNT	Destination Wrap Count Register	Section 11.8.17
0x102F	DST_WRAP_STEP	Destination Wrap Step Size Register	Section 11.8.18
0x1030	SRC_BEG_ADDR_SHADOW	Shadow Source Begin and Current Address Pointer Registers	Section 11.8.19
0x1032	SRC_ADDR_SHADOW		Section 11.8.19
0x1034	SRC_BEG_ADDR	Active Source Begin and Current Address Pointer Registers	Section 11.8.20
0x1036	SRC_ADDR		Section 11.8.20
0x1038	DST_BEG_ADDR_SHADOW	Shadow Destination Begin and Current Address Pointer Registers	Section 11.8.21

(1) All DMA register writes are EALLOW protected.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 11-2. DMA Register Summary[(1)] (continued)

Address	Acronym	Description	Section
0x103A	DST_ADDR_SHADOW		Section 11.8.21
0x103C	DST_BEG_ADDR	Active Destination Begin and Current Address Pointer Registers	Section 11.8.22
0x103E	DST_ADDR		Section 11.8.22
0x103F	Reserved	Reserved	
DMA Channel 2 Registers			
0x1040 0x105F	Same as above		
DMA Channel 3 Registers			
0x1060 0x107F	Same as above		
DMA Channel 4 Registers			
0x1080 0x109F	Same as above		
DMA Channel 5 Registers			
0x10A0 0x10BF	Same as above		
DMA Channel 6 Registers			
0x10C0 0x10DF	Same as above		

11.8.1 DMA Control Register (DMACTRL) — EALLOW Protected

The DMA control register (DMACTRL) is shown in Figure 11-8 and described in Table 11-3.

Figure 11-8. DMA Control Register (DMACTRL)

15	8
Reserved	
R-0	

7	2	1	0
Reserved		PRIORITY RESET	HARD RESET
R-0		R0/S-0	R0/S-0

LEGEND: R0/S = Read 0/Set; R = Read only; -*n* = value after reset

Table 11-3. DMA Control Register (DMACTRL) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved		Reserved
1	PRIORITYRESET	0	The priority reset bit resets the round-robin state machine when a 1 is written. Service starts from the first enabled channel. Writes of 0 are ignored and this bit always reads back a 0. When a 1 is written to this bit, any pending burst transfer completes before resetting the channel priority machine. If CH1 is configured as a high priority channel, and this bit is written to while CH1 is servicing a burst, the CH1 burst is completed and then any lower priority channel burst is also completed (if CH1 interrupted in the middle of a burst), before the state machine is reset. In case CH1 is high priority, the *state machine* restarts from CH2 (or the next highest enabled channel).

Copyright © 2012–2019, Texas Instruments Incorporated

Table 11-3. DMA Control Register (DMACTRL) Field Descriptions (continued)

Bit	Field	Value	Description
0	HARDRESET	0	Writing a 1 to the hard reset bit resets the whole DMA and aborts any current access (similar to applying a device reset). Writes of 0 are ignored and this bit always reads back a 0. For a *soft* reset, a bit is provided for each channel to perform a gentler reset. Refer to the channel control registers. If the DMA was performing an access to the XINTF and the DMA access was stalled (XREADY not responding), then a HARDRESET would abort the access. The XINTF access would only complete if XREADY is released. When writing to this bit, there is a one cycle delay before it takes effect. Hence at least a one cycle delay (i.e., a NOP instruction) after writing to this bit should be introduced before attempting an access to any other DMA register.

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.2 Debug Control Register (DEBUGCTRL) — EALLOW Protected

The debug control register (DEBUGCTRL) is shown in Figure 11-9 and described in Table 11-4.

Figure 11-9. Debug Control Register (DEBUGCTRL)

15	14	0
FREE	Reserved	
R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-4. Debug Control Register (DEBUGCTRL) Field Descriptions

Bit	Field	Value	Description
15	FREE		Emulation Control Bit: This bit specifies the action when an emulation halt event occurs.
		0	DMA runs until the current DMA read-write access is completed and the current status of a DMA is frozen. See the *HALT* points in Figure 11-6 for possible halt states.
		1	DMA is unaffected by emulation suspend (run free)
14-0	Reserved		Reserved

11.8.3 Revision Register (REVISION)

The revision register (REVISION) is shown in Figure 11-10 and described in Table 11-5.

Figure 11-10. Revision Register (REVISION)

15	8	7	0
TYPE		REV	
R		R	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-5. Revision Register (REVISION) Field Descriptions

Bit	Field	Value	Description
15-8	TYPE		DMA Type Bits. A type change represents a major functional feature difference in a peripheral module. Within a peripheral type, there may be minor differences between devices which do not affect the basic functionality of the module. These device-specific differences are listed in the *TMS320x28xx, 28xxx DSP Peripheral Reference Guide* (SPRU566).
		0x0000	This document describes a Type0 DMA.
7-0	REV		DMA Silicon Revision Bits: These bits specify the DMA revision and are changed if any bug fixes are performed.
		0x0000	First release

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.4 Priority Control Register 1 (PRIORITYCTRL1) — EALLOW Protected

The priority control register 1 (PRIORITYCTRL1) is shown in Figure 11-11 and described in Table 11-6.

Figure 11-11. Priority Control Register 1 (PRIORITYCTRL1)

15	1	0
Reserved		CH1 PRIORITY
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-6. Priority Control Register 1 (PRIORITYCTRL1) Field Descriptions

Bit	Field	Value	Description
15-1	Reserved		Reserved
0	CH1PRIORITY		DMA Ch1 Priority: This bit selects whether channel 1 has higher priority or not:
		0	Same priority as all other channels
		1	Highest priority channel
			Channel priority can only be changed when all channels are disabled. A priority reset should be performed before restarting channels after changing priority.

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.5 Priority Status Register (PRIORITYSTAT)

The priority status register (PRIORITYSTAT) is shown in Figure 11-12 and described in Table 11-7.

Figure 11-12. Priority Status Register (PRIORITYSTAT)

15					8
Reserved					
R-0					

7	6	4	3	2	0
Reserved	ACTIVESTS_SHADOW		Reserved	ACTIVESTS	
R-0	R-0		R-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-7. Priority Status Register (PRIORITYSTAT) Field Descriptions

Bit	Field	Value	Description
15-7	Reserved		Reserved
6-4	ACTIVESTS_SHADOW		Active Channel Status Shadow Bits: These bits are only useful when CH1 is enabled as a higher priority channel. When CH1 is serviced, the ACTIVESTS bits are copied to the shadow bits and indicate which channel was interrupted by CH1. When CH1 service is completed, the shadow bits are copied back to the ACTIVESTS bits. If this bit field is zero or the same as the ACTIVESTS bit field, then no channel is pending due to a CH1 interrupt. When CH1 is not a higher priority channel, these bits should be ignored:
		0,0,0	No channel pending
		0,0,1	CH 1
		0,1,0	CH 2
		0,1,1	CH 3
		1,0,0	CH 4
		1,0,1	CH 5
		1,1,0	CH 6
3	Reserved		Reserved
2-0	ACTIVESTS		Active Channel Status Bits: These bits indicate which channel is currently active or performing a transfer:
		0,0,0	no channel active
		0,0,1	CH 1
		0,1,0	CH 2
		0,1,1	CH 3
		1,0,0	CH 4
		1,0,1	CH 5
		1,1,0	CH 6

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.6 Mode Register (MODE) — EALLOW Protected

The mode register (MODE) is shown in Figure 11-13 and described in Table 11-8.

Figure 11-13. Mode Register (MODE)

15	14	13	12	11	10	9	8
CHINTE	DATASIZE	Reserved		CONTINUOUS	ONESHOT	CHINTMODE	PERINTE
R/W-0	R/W-0	R/W-0		R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4				0
OVRINTE	Reserved		PERINTSEL				
R/W-0	R-0		R/W-0				

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-8. Mode Register (MODE) Field Descriptions

Bit	Field	Value	Description
15	CHINTE		Channel Interrupt Enable Bit: This bit enables/disables the respective DMA channel interrupt to the CPU (via the PIE).
		0	Interrupt disabled
		1	Interrupt enabled
14	DATASIZE		Data Size Mode Bit: This bit selects if the DMA channel transfers 16-bits or 32-bits of data at a time.
		0	16-bit data transfer size
		1	32-bit data transfer size **NOTE:** Regardless of the value of this bit all of the registers in the DMA refer to 16-bit words. The only effect this bit causes is whether the databus width is 16 or 32 bits. It is up to you to configure the pointer step increment and size to accommodate 32-bit data transfers. See section Section 11.6 for details.
13-12	Reserved		Reserved
11	CONTINUOUS		Continuous Mode Bit: If this bit is set to 1, then DMA re-initializes when TRANSFER_COUNT is zero and waits for the next interrupt event trigger. If this bit is 0, then the DMA stops and clears the RUNSTS bit to 0.
10	ONESHOT		One Shot Mode Bit: If this bit is set to 1, then subsequent burst transfers occur without additional event triggers after the first event trigger. If this bit is 0 then only one burst transfer is performed per event trigger. **Note:** High-priority mode and One-shot mode may not be used at the same time on CH1.
9	CHINTMODE		Channel Interrupt Generation Mode Bit: This bit specifies when the respective DMA channel interrupt should be generated to the CPU (via the PIE).
		0	Generate interrupt at beginning of new transfer
		1	Generate interrupt at end of transfer.
8	PERINTE		Peripheral Interrupt Trigger Enable Bit: This bit enables/disables the selected peripheral interrupt trigger to the DMA.
		0	Interrupt trigger disabled. Neither the selected peripheral nor software can start a DMA burst.
		1	Interrupt trigger enabled.
7	OVRINTE		Overflow Interrupt Enable: This bit when set to 1 enables the DMA to generate an interrupt when an overflow event is detected.
		0	Overflow interrupt disabled
		1	Overflow interrupt enabled
			An overflow interrupt is generated when the PERINTFLG is set and another interrupt event occurs. The PERINTFLG being set indicates a previous peripheral event is latched and has not been serviced by the DMA.
6-5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 11-8. Mode Register (MODE) Field Descriptions (continued)

Bit	Field	Value	Description		
4-0	PERINTSEL		Peripheral Interrupt Source Select Bits: These bits select which interrupt triggers a DMA burst for the given channel. Only one interrupt source can be selected. A DMA burst can also be forced via the PERINTFRC bit.		
		Value	**Interrupt**	**Sync**	**Peripheral**
		0	None	None	No peripheral connection
		1	ADCINT1	None	ADC
		2	ADCINT2	None	
		3	XINT1	None	External Interrupts
		4	XINT2	None	
		5	XINT3	None	
		6	Reserved	None	No peripheral connection
		7	ePWM8SOCA	None	ePWM8
		8	ePWM8SOCB	None	
		9	ePWM9SOCA	None	ePWM9
		10	ePWM9SOCB	None	
		11	TINT0	None	CPU Timers
		12	TINT1	None	
		13	TINT2	None	
		14	MXEVTA	None	McBSP-A
		15	MREVTA	None	
		16	ADCINT3	None	ADC
		17	ADCINT4	None	
		18	ePWM1SOCA	None	ePWM1
		19	ePWM1SOCB	None	
		20	ePWM2SOCA	None	ePWM2
		21	ePWM2SOCB	None	
		22	ePWM3SOCA	None	ePWM3
		23	ePWM3SOCB	None	
		24	ePWM4SOCA	None	ePWM4
		25	ePWM4SOCB	None	
		26	ePWM5SOCA	None	ePWM5
		27	ePWM5SOCB	None	
		28	ePWM6SOCA	None	ePWM6
		29	ePWM6SOCB	None	
		30	ePWM7SOCA	None	ePWM7
		31	ePWM7SOCB	None	

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.7 Control Register (CONTROL) — EALLOW Protected

The control register (CONTROL) is shown in Figure 11-14 and described in Table 11-9.

Figure 11-14. Control Register (CONTROL)

15	14	13	12	11	10	9	8
Reserved	OVRFLG	RUNSTS	BURSTSTS	TRANSFERSTS	Reserved		PERINTFLG
R-0	R-0	R-0	R-0	R-0	R-0		R-0
7	6	5	4	3	2	1	0
ERRCLR	Reserved		PERINTCLR	PERINTFRC	SOFTRESET	HALT	RUN
R0/S-0	R-0		R0/S-0	R0/S-0	R0/S-0	R0/S-0	R0/S-0

LEGEND: R0/S = Read 0/Set; R = Read only; -*n* = value after reset

Table 11-9. Control Register (CONTROL) Field Descriptions

Bit	Field	Value	Description
15	Reserved		Reserved
14	OVRFLG		Overflow Flag Bit: This bit indicates if a peripheral interrupt event trigger is received from the selected peripheral and the PERINTFLG is already set.
		0	No overflow event
		1	Overflow event
			The ERRCLR bit can be used to clear the state of this bit to 0. The OVRFLG bit is not affected by the PERINTFRC event.
13	RUNSTS		Run Status Bit: This bit is set to 1 when the RUN bit is written to with a 1. This indicates the DMA channel is now ready to process peripheral interrupt event triggers. This bit is cleared to 0 when TRANSFER_COUNT reaches zero and CONTINUOUS mode bit is set to 0. This bit is also cleared to 0 when either the HARDRESET bit, the SOFTRESET bit, or the HALT bit is activated.
		0	Chanel is disabled.
		1	Channel is enabled.
12	BURSTSTS		Burst Status Bit: This bit is set to 1 when a DMA burst transfer begins and the BURST_COUNT is initialized with the BURST_SIZE. This bit is cleared to zero when BURST_COUNT reaches zero. This bit is also cleared to 0 when either the HARDRESET or the SOFTRESET bit is activated.
		0	No burst activity
		1	The DMA is currently servicing or suspending a burst transfer from this channel.
11	TRANSFERSTS		Transfer Status Bit: This bit is set to 1 when a DMA transfer begins and the address registers are copied to the shadow set and the TRANSFER_COUNT is initialized with the TRANSFER_SIZE. This bit is cleared to zero when TRANSFER_COUNT reaches zero. This bit is also cleared to 0 when either the HARDRESET or the SOFTRESET bit is activated.
		0	No transfer activity
		1	The channel is currently in the middle of a transfer regardless of whether a burst of data is actively being transferred or not.
10-9	Reserved		Reserved
8	PERINTFLG		Peripheral Interrupt Trigger Flag Bit: This bit indicates if a peripheral interrupt event trigger has occurred. This flag is automatically cleared when the first burst transfer begins.
		0	No interrupt event trigger
		1	Interrupt event trigger
			The PERINTFRC bit can be used to set the state of this bit to 1 and force a software DMA event. The PERINTCLR bit can be used to clear the state of this bit to 0.
7	ERRCLR	0	Error Clear Bit: Writing a 1 to this bit will clear any latched sync error event and clear the SYNCERR bit. This bit will also clear the OVRFLG bit. This bit would normally be used when initializing the DMA for the first time or if an overflow condition is detected. If an ADCSYNC error event or overflow event occurs at the same time as writing to this bit, the ADC or overrun has priority and the SYNCERR or OVRFLG bit is set.
6-5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 11-9. Control Register (CONTROL) Field Descriptions (continued)

Bit	Field	Value	Description
4	PERINTCLR	0	Peripheral Interrupt Clear Bit: Writing a 1 to this bit clears any latched peripheral interrupt event and clears the PERINTFLG bit. This bit would normally be used when initializing the DMA for the first time. If a peripheral event occurs at the same time as writing to this bit, the peripheral has priority and the PERINTFLG bit is set.
3	PERINTFRC	0	Peripheral Interrupt Force Bit: Writing a 1 to this bit latches a peripheral interrupt event trigger and sets the PERINTFLG bit. If the PERINTE bit is set, this bit can be used like a software force for a DMA burst transfer.
2	SOFTRESET	0	Channel Soft Reset Bit: Writing a 1 to this bit completes current read-write access and places the channel into a default state as follows: RUNSTS = 0 TRANSFERSTS = 0 BURSTSTS = 0 BURST_COUNT = 0 TRANSFER_COUNT = 0 SRC_WRAP_COUNT = 0 DST_WRAP_COUNT = 0 This is a *soft* reset that basically allows the DMA to complete the current read-write access and then places the DMA channel into the default reset state.
1	HALT	0	Channel Halt Bit: Writing a 1 to this bit halts the DMA at the current state and any current read-write access is completed. See Figure 11-6 for the various positions the state machine can be at when HALTED. The RUNSTS bit is set to 0. To take the device out of HALT, the RUN bit needs to be activated.
0	RUN	0	Channel Run Bit: Writing a 1 to this bit starts the DMA channel. The RUNSTS bit is set to 1. This bit is also used to take the device out of HALT. The RUN bit is typically used to start the DMA running after you have configured the DMA. It will then wait for the first interrupt event (PERINTFLG == 1) to start operation. The RUN bit can also be used to take the DMA channel out of a HALT condition See Figure 11-6 for the various positions the state machine can be at when HALTED.

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.8 Burst Size Register (BURST_SIZE) — EALLOW Protected

The burst size register (BURST_SIZE) is shown in Figure 11-15 and described in Table 11-10.

Figure 11-15. Burst Size Register (BURST_SIZE)

15	5	4	0
Reserved		BURSTSIZE	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-10. Burst Size Register (BURST_SIZE) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved		Reserved
4-0	BURSTSIZE		These bits specify the burst transfer size:
		0	Transfer 1 word in a burst
		1	Transfer 2 words in a burst
		...	...
		31	Transfer 32 words in a burst

11.8.9 BURST_COUNT Register

The burst count register (BURST_COUNT) is shown in Figure 11-16 and described in Table 11-11.

Figure 11-16. Burst Count Register (BURST_COUNT)

15	5	4	0
Reserved		BURSTCOUNT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-11. Burst Count Register (BURST_COUNT) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved		Reserved
4-0	BURSTCOUNT		These bits indicate the current burst counter value:
		0	0 word left in a burst
		1	1 word left in a burst
		2	2 words left in a burst
		...	...
		31	31 words left in a burst
			The above values represent the state of the counter at the HALT conditions.

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.10 Source Burst Step Register Size (SRC_BURST_STEP) — EALLOW Protected

The source burst step size register (SRC_BURST_STEP) is shown in Figure 11-17 and described in Table 11-12.

Figure 11-17. Source Burst Step Size Register (SRC_BURST_STEP)

15	0
SRCBURSTSTEP	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-12. Source Burst Step Size Register (SRC_BURST_STEP) Field Descriptions

Bit	Field	Value	Description
15-0	SRCBURSTSTEP		These bits specify the source address post-increment/decrement step size while processing a burst of data:
		0x0FFF	Add 4095 to address
		...	...
		0x0002	Add 2 to address
		0x0001	Add 1 to address
		0x0000	No address change
		0xFFFF	Sub 1 from address
		0xFFFE	Sub 2 from address
		...	...
		0xF000	Sub 4096 from address
			Only values from -4096 to 4095 are valid.

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.11 Destination Burst Step Register Size (DST_BURST_STEP) — EALLOW Protected

The destination burst step register size (DST_BURST_STEP) is shown in Figure 11-18 and described in Table 11-13.

Figure 11-18. Destination Burst Step Register Size (DST_BURST_STEP)

15	0
DSTBURSTSTEP	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-13. Destination Burst Step Register Size (DST_BURST_STEP) Field Descriptions

Bit	Field	Value	Description
15-0	DSTBURSTSTEP		These bits specify the destination address post-increment/decrement step size while processing a burst of data:
		0x0FFF	Add 4095 to address
		...	...
		0x0002	Add 2 to address
		0x0001	Add 1 to address
		0x0000	No address change
		0xFFFF	Sub 1 from address
		0xFFFE	Sub 2 from address
		...	...
		0xF000	Sub 4096 from address
			Only values from -4096 to 4095 are valid.

11.8.12 Transfer Size Register (TRANSFER_SIZE) — EALLOW Protected

The transfer size register (TRANSFER_SIZE) is shown in Figure 11-19 and described in Table 11-14.

Figure 11-19. Transfer Size Register (TRANSFER_SIZE)

15	0
TRANSFERSIZE	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-14. Transfer Size Register (TRANSFER_SIZE) Field Descriptions

Bit	Field	Value	Description
15-0	TRANSFERSIZE		These bits specify the number of bursts to transfer:
		0x0000	Transfer 1 burst
		0x0001	Transfer 2 bursts
		0x0002	Transfer 3 bursts
		...	...
		0xFFFF	Transfer 65536 bursts

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.13 Transfer Count Register (TRANSFER_COUNT)

The transfer count register (TRANSFER_COUNT) is shown in Figure 11-20 and described in Table 11-15.

Figure 11-20. Transfer Count Register (TRANSFER_COUNT)

15	0
TRANSFERCOUNT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-15. Transfer Count Register (TRANSFER_COUNT) Field Descriptions

Bit	Field	Value	Description
15-0	TRANSFERCOUNT		These bits specify the current transfer counter value:
		0x0000	0 bursts left to transfer
		0x0001	1 burst left to transfer
		0x0002	2 bursts left to transfer
		...	...
		0xFFFF	65535 bursts left to transfer
			The above values represent the state of the counter at the HALT conditions.

11.8.14 Source Transfer Step Size Register (SRC_TRANSFER_STEP) — EALLOW Protected

The source transfer step size register (SRC_TRANSFER_STEP) is shown in Figure 11-21 and described in Table 11-16.

Figure 11-21. Source Transfer Step Size Register (SRC_TRANSFER_STEP)

15	0
SRCTRANSFERSTEP	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-16. Source Transfer Step Size Register (SRC_TRANSFER_STEP) Field Descriptions

Bit	Field	Value	Description
15-0	SRCTRANSFERSTEP		These bits specify the source address pointer post-increment/decrement step size after processing a burst of data:
		0x0FFF	Add 4095 to address
		...	...
		0x0002	Add 2 to address
		0x0001	Add 1 to address
		0x0000	No address change
		0xFFFF	Sub 1 from address
		0xFFFE	Sub 2 from address
		...	...
		0xF000	Sub 4096 from address
			Only values from -4096 to 4095 are valid.

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.15 Destination Transfer Step Size Register (DST_TRANSFER_STEP) — EALLOW Protected

The destination transfer step size register (DST_TRANSFER_STEP) is shown in Figure 11-22 and described in Table 11-17.

Figure 11-22. Destination Transfer Step Size Register (DST_TRANSFER_STEP)

15	0
DSTTRANSFERSTEP	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-17. Destination Transfer Step Size Register (DST_TRANSFER_STEP) Field Descriptions

Bit	Field	Value	Description
15-0	DSTTRANSFERSTEP		These bits specify the destination address pointer post-increment/decrement step size after processing a burst of data:
		0x0FFF	Add 4095 to address
		...	...
		0x0002	Add 2 to address
		0x0001	Add 1 to address
		0x0000	No address change
		0xFFFF	Sub 1 from address
		0xFFFE	Sub 2 from address
		...	...
		0xF000	Sub 4096 from address
			Only values from -4096 to 4095 are valid.

11.8.16 Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE) — EALLOW protected)

The source/destination wrap size register is shown in Figure 11-23 and described in Table 11-18.

Figure 11-23. Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE)

15	0
WRAPSIZE	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-18. Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE) Field Descriptions

Bit	Field	Value	Description
15-0	WRAPSIZE		These bits specify the number of bursts to transfer before wrapping back to begin address pointer:
		0x0000	Wrap after 1 burst
		0x0001	Wrap after 2 bursts
		0x0002	Wrap after 3 bursts
		...	...
		0xFFFF	Wrap after 65536 bursts
			To *disable* the wrap function, set the WRAPSIZE bit field to a number larger than the TRANSFERSIZE bit field.

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.17 Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT)

The source/destination wrap count register (SCR/DST_WRAP_COUNT) is shown in Figure 11-24 and described in Table 11-19.

Figure 11-24. Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT)

15	0
WRAPCOUNT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-19. Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT) Field Descriptions

Bit	Field	Value	Description
15-0	WRAPCOUNT		These bits indicate the current wrap counter value:
		0x0000	Wrap complete
		0x0001	1 burst left
		0x0002	2 burst left
		...	...
		0xFFFF	65535 burst left
			The above values represent the state of the counter at the HALT conditions.

11.8.18 Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP) — EALLOW Protected

The source/destination wrap step size register (SRC/DST_WRAP_STEP) are shown in Figure 11-25 and described in Table 11-20.

Figure 11-25. Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP)

15	0
WRAPSTEP	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-20. Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP) Field Descriptions

Bit	Field	Value	Description
15-0	WRAPSTEP		These bits specify the source begin address pointer post-increment/decrement step size after wrap counter expires:
		0x0FFF	Add 4095 to address
		...	...
		0x0002	Add 2 to address
		0x0001	Add 1 to address
		0x0000	No address change
		0xFFFF	Sub 1 from address
		0xFFFE	Sub 2 from address
		...	...
		0xF000	Sub 4096 from address
			Only values from -4096 to 4095 are valid.

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.19 Shadow Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW) — All EALLOW Protected

The shadow source begin and current address pointer registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW) are shown in Figure 11-26 and described in Table 11-21.

Figure 11-26. Shadow Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW)

31	22	21	0
Reserved		BEGADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-21. Shadow Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved		Reserved
21-0	BEGADDR		22-bit address value

11.8.20 Active Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR/DST_BEG_ADDR)

The active source begin and current address pointer registers (SRC_BEG_ADDR/DST_BEG_ADDR) are shown in Table 11-22 and described in Table 11-22.

Figure 11-27. Active Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR/DST_BEG_ADDR)

31	22	21	0
Reserved		BEGADDR	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-22. Active Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR/DST_BEG_ADDR) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved		Reserved
21-0	BEGADDR		22-bit address value

Copyright © 2012–2019, Texas Instruments Incorporated

11.8.21 Shadow Destination Begin and Current Address Pointer Registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW) — All EALLOW Protected

The shadow destination begin and current address pointer registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW) are shown in Figure 11-28 and described in Table 11-23.

Figure 11-28. Shadow Destination Begin and Current Address Pointer Registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW)

31	22	21	0
Reserved		ADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-23. Shadow Destination Begin and Current Address Pointer Registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved		Reserved
21-0	ADDR		22-bit address value

11.8.22 Active Destination Begin and Current Address Pointer Registers (SRC_ADDR/DST_ADDR)

The active destination begin and current address pointer registers (SRC_ADDR/DST_ADDR) are shown in Figure 11-29 and described in Table 11-24.

Figure 11-29. Active Destination Begin and Current Address Pointer Registers (SRC_ADDR/DST_ADDR)

31	22	21	0
Reserved		ADDR	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11-24. Active Destination Begin and Current Address Pointer Registers (SRC_ADDR/DST_ADDR) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved		Reserved
21-0	ADDR		22-bit address value

Copyright © 2012–2019, Texas Instruments Incorporated

C28 Serial Peripheral Interface (SPI)

This chapter describes the serial peripheral interface (SPI).The serial peripheral interface (SPI) is a high-speed synchronous serial input/ output (I/O) port that allows a serial bit stream of programmed length (one to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communications between the DSP controller and external peripherals or another controller. Typical applications include external I/O or peripheral expansion via devices such as shift registers, display drivers, and analog-to-digital converters (ADCs). Multi-device communications are supported by the master/slave operation of the SPI. The port supports a 4 -level, receive and transmit FIFO for reducing CPU servicing overhead.

Topic **Page**

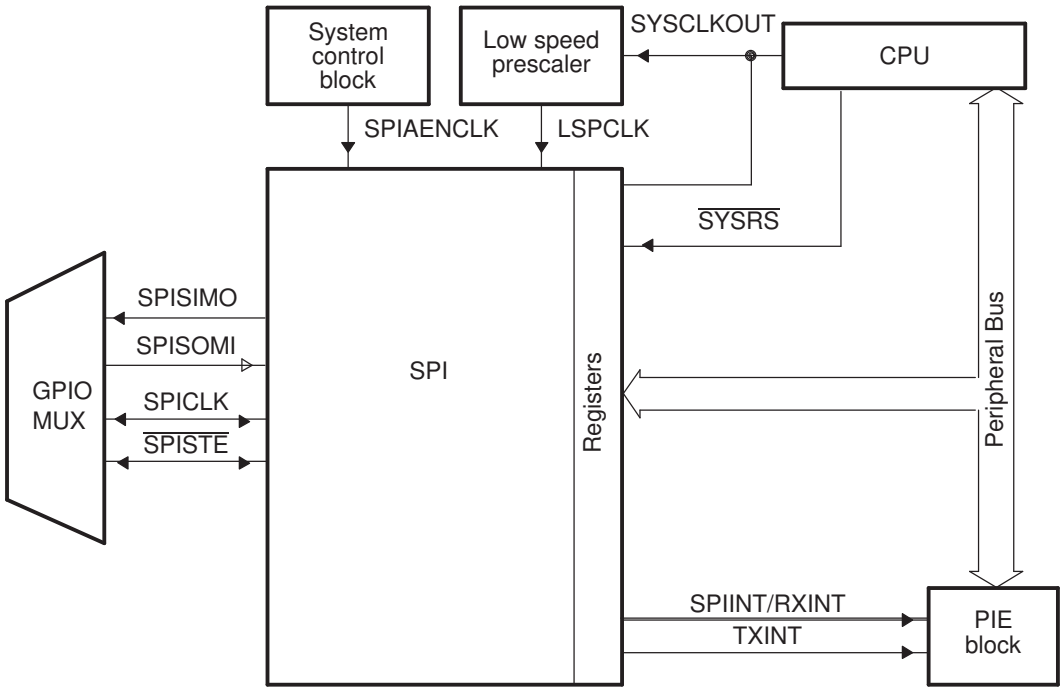
12.1 Enhanced SPI Module Overview **945**
12.2 C28 SPI-A to M3 SSI3 Internal Loopback **961**
12.3 SPI Registers and Waveforms **964**

Copyright © 2012–2019, Texas Instruments Incorporated

12.1 Enhanced SPI Module Overview

Figure 12-1 shows the SPI CPU interfaces.

Figure 12-1. SPI CPU Interface



The SPI module features include:

- SPISOMI: SPI slave-output/master-input pin
- SPISIMO: SPI slave-input/master-output pin
- $\overline{\text{SPISTE}}$: SPI slave transmit-enable pin
- SPICLK: SPI serial-clock pin

NOTE: All four pins can be used as GPIO, if the SPI module is not used.

- Two operational modes: master and slave
- Baud rate: 125 different programmable rates. The maximum baud rate that can be employed is limited by the maximum speed of the I/O buffers used on the SPI pins. See the device-specific data sheet for more details.
- Data word length: one to sixteen data bits
-
- Four clocking schemes (controlled by clock polarity and clock phase bits) include:
  - Falling edge without phase delay: SPICLK active-high. SPI transmits data on the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
  - Falling edge with phase delay: SPICLK active-high. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge without phase delay: SPICLK inactive-low. SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge with phase delay: SPICLK inactive-low. SPI transmits data one half-cycle ahead of the rising edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
- Simultaneous receive and transmit operation (transmit function can be disabled in software)
- Transmitter and receiver operations are accomplished through either interrupt- driven or polled algorithms.

Copyright © 2012–2019, Texas Instruments Incorporated

- 12 SPI module control registers: Located in the control register frame beginning at address 7040h.

NOTE: All registers in this module are 16-bit registers that are connected to Peripheral Frame 2. When a register is accessed, the register data is in the lower byte (7−0), and the upper byte (15−8) is read as zeros. Writing to the upper byte has no effect.

Enhanced Features:

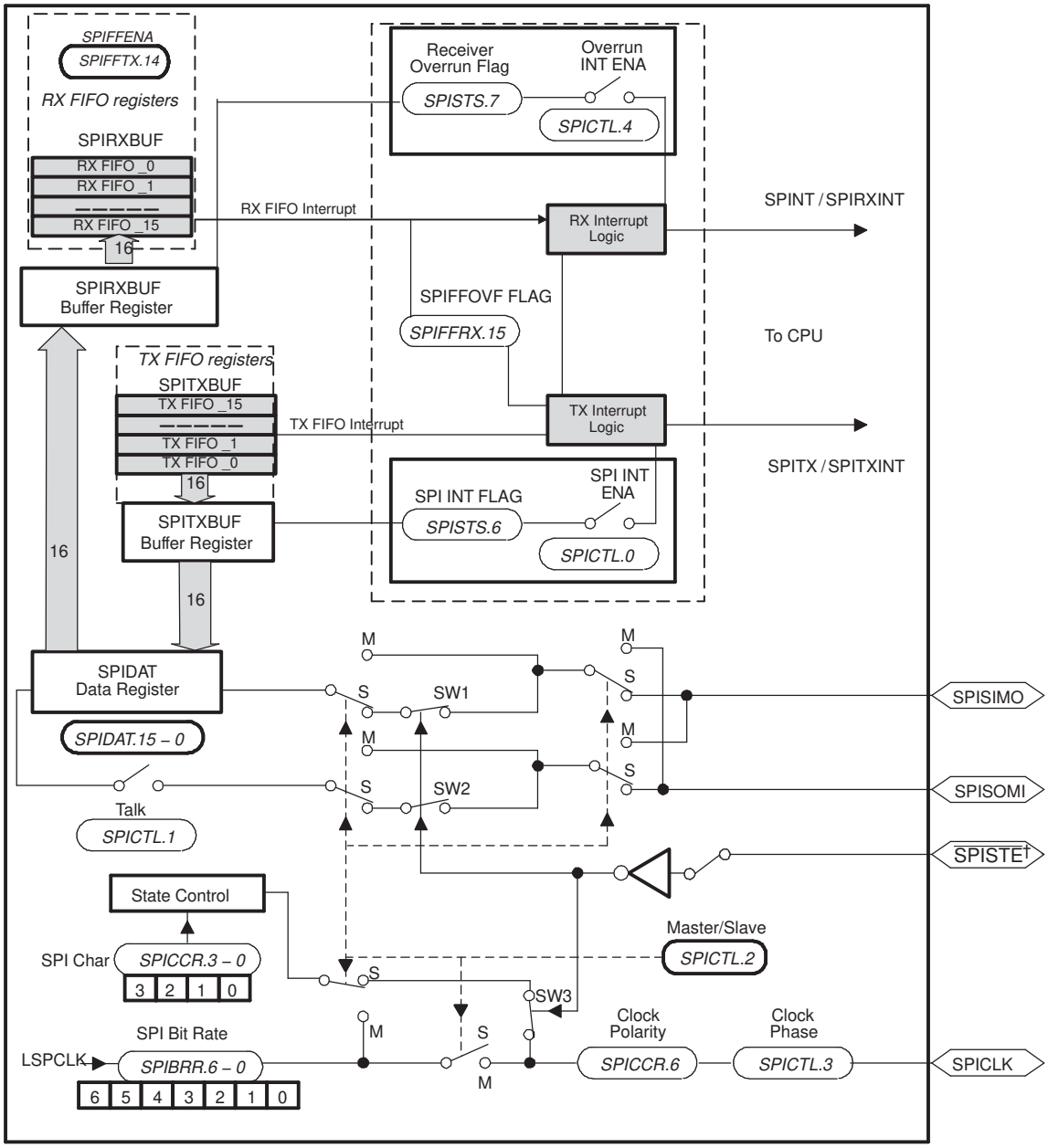
- 4 -level transmit/receive FIFO
- Delayed transmit control

12.1.1 SPI Block Diagram

Figure 12-2 is a block diagram of the SPI in slave mode, showing the basic control blocks available on the SPI module.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-2. Serial Peripheral Interface Module Block Diagram



A $\overline{\text{SPISTE}}$ of a slave device is driven low by the master.

Copyright © 2012–2019, Texas Instruments Incorporated

12.1.2 SPI Module Signal Summary

Table 12-1. SPI Module Signal Summary

Signal Name	Description
External Signals	
SPICLK	SPI clock
SPISIMO	SPI slave in, master out
SPISOMI	SPI slave out, master in
$\overline{\text{SPISTE}}$	SPI slave transmit enable
Control	
SPI Clock Rate	LSPCLK
Interrupt signals	
SPIRXINT	Transmit interrupt/ Receive Interrupt in non FIFO mode (referred to as SPI INT)
	Receive in interrupt in FIFO mode
SPITXINT	Transmit interrupt – FIFO

12.1.3 Overview of SPI Module Registers

The SPI port operation is configured and controlled by the registers listed in Table 12-2.

Table 12-2. SPI Registers

Name	Address Range	Size (x16)	Description
SPICCR	0x0000-7040	1	SPI Configuration Control Register
SPICTL	0x0000-7041	1	SPI Operation Control Register
SPIST	0x0000-7042	1	SPI Status Register
SPIBRR	0x0000-7044	1	SPI Baud Rate Register
SPIEMU	0x0000-7046	1	SPI Emulation Buffer Register
SPIRXBUF	0x0000-7047	1	SPI Serial Input Buffer Register
SPITXBUF	0x0000-7048	1	SPI Serial Output Buffer Register
SPIDAT	0x0000-7049	1	SPI Serial Data Register
SPIFFTX	0x0000-704A	1	SPI FIFO Transmit Register
SPIFFRX	0x0000-704B	1	SPI FIFO Receive Register
SPIFFCT	0x0000-704C	1	SPI FIFO Control Register
SPIPRI	0x0000-704F	1	SPI Priority Control Register

This SPI has 16-bit transmit and receive capability, with double-buffered transmit and double-buffered receive. All data registers are 16-bits wide.

The SPI is no longer limited to a maximum transmission rate of LSPCLK/8 in slave mode. The maximum transmission rate in both slave mode and master mode is now LSPCLK/4.

Writes of transmit data to the serial data register, SPIDAT (and the new transmit buffer, SPITXBUF), must be left-justified within a 16-bit register.

The control and data bits for general-purpose bit I/O multiplexing have been removed from this peripheral, along with the associated registers, SPIPC1 (704Dh) and SPIPC2 (704Eh). These bits are now in the General-Purpose I/O registers.

Twelve registers inside the SPI module control the SPI operations:

- SPICCR (SPI configuration control register). Contains control bits used for SPI configuration
  - SPI module software reset
  - SPICLK polarity selection
  - Four SPI character-length control bits

Copyright © 2012–2019, Texas Instruments Incorporated

- SPICTL (SPI operation control register). Contains control bits for data transmission
  - Two SPI interrupt enable bits
  - SPICLK phase selection
  - Operational mode (master/slave)
  - Data transmission enable
- SPISTS (SPI status register). Contains two receive buffer status bits and one transmit buffer status bit
  - RECEIVER OVERRUN
  - SPI INT FLAG
  - TX BUF FULL FLAG
- SPIBRR (SPI baud rate register). Contains seven bits that determine the bit transfer rate
- SPIRXEMU (SPI receive emulation buffer register). Contains the received data. This register is used for emulation purposes only. The SPIRXBUF should be used for normal operation
- SPIRXBUF (SPI receive buffer — the serial receive buffer register). Contains the received data
- SPITXBUF (SPI transmit buffer — the serial transmit buffer register). Contains the next character to be transmitted
- SPIDAT (SPI data register). Contains data to be transmitted by the SPI, acting as the transmit/receive shift register. Data written to SPIDAT is shifted out on subsequent SPICLK cycles. For every bit shifted out of the SPI, a bit from the receive bit stream is shifted into the other end of the shift register
- SPIPRI (SPI priority register). Contains bits that specify interrupt priority and determine SPI operation on the XDS™ emulator during program suspensions.

12.1.4 SPI Operation

This section describes the operation of the SPI. Included are explanations of the operation modes, interrupts, data format, clock sources, and initialization. Typical timing diagrams for data transfers are given.

12.1.4.1 Introduction to Operation

Figure 12-3 shows typical connections of the SPI for communications between two controllers: a master and a slave.

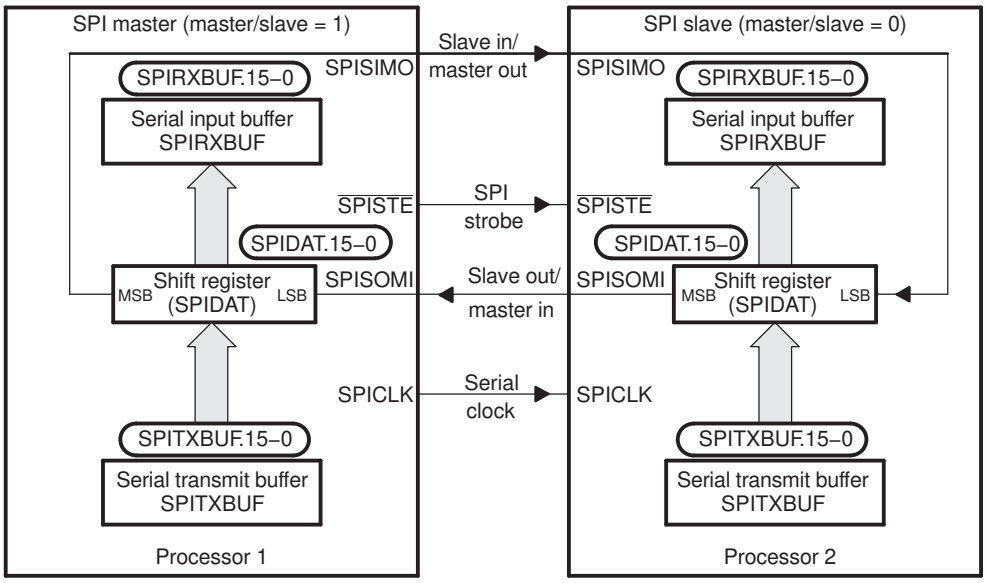
The master initiates data transfer by sending the SPICLK signal. For both the slave and the master, data is shifted out of the shift registers on one edge of the SPICLK and latched into the shift register on the opposite SPICLK clock edge. If the CLOCK PHASE bit (SPICTL.3) is high, data is transmitted and received a half-cycle before the SPICLK transition (see Section 12.1.4.2). As a result, both controllers send and receive data simultaneously. The application software determines whether the data is meaningful or dummy data. There are three possible methods for data transmission:

- Master sends data; slave sends dummy data.
- Master sends data; slave sends data.
- Master sends dummy data; slave sends data.

The master can initiate data transfer at any time because it controls the SPICLK signal. The software, however, determines how the master detects when the slave is ready to broadcast data.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-3. SPI Master/Slave Connection



12.1.4.2 SPI Module Slave and Master Operation Modes

The SPI can operate in master or slave mode. The MASTER/SLAVE bit (SPICTL.2) selects the operating mode and the source of the SPICLK signal.

12.1.4.2.1 Master Mode

In the master mode (MASTER/SLAVE = 1), the SPI provides the serial clock on the SPICLK pin for the entire serial communications network. Data is output on the SPISIMO pin and latched from the SPISOMI pin.

The SPIBRR register determines both the transmit and receive bit transfer rate for the network. SPIBRR can select 125 different data transfer rates.

Data written to SPIDAT or SPITXBUF initiates data transmission on the SPISIMO pin, MSB (most significant bit) first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB (least significant bit) of SPIDAT. When the selected number of bits has been transmitted, the received data is transferred to the SPIRXBUF (buffered receiver) for the CPU to read. Data is stored right-justified in SPIRXBUF.

When the specified number of data bits has been shifted through SPIDAT, the following events occur:

- SPIDAT contents are transferred to SPIRXBUF.
- SPI INT FLAG bit (SPISTS.6) is set to 1.
- If there is valid data in the transmit buffer SPITXBUF, as indicated by the TXBUF FULL bit in SPISTS, this data is transferred to SPIDAT and is transmitted; otherwise, SPICLK stops after all bits have been shifted out of SPIDAT.
- If the SPI INT ENA bit (SPICTL.0) is set to 1, an interrupt is asserted.

In a typical application, the $\overline{\text{SPISTE}}$ pin serves as a chip-enable pin for a slave SPI device. This pin is driven low by the master before transmitting data to the slave and is taken high after the transmission is complete.

12.1.4.2.2 Slave Mode

In the slave mode (MASTER/SLAVE = 0), data shifts out on the SPISOMI pin and in on the SPISIMO pin. The SPICLK pin is used as the input for the serial shift clock, which is supplied from the external network master. The transfer rate is defined by this clock. The SPICLK input frequency should be no greater than the LSPCLK frequency divided by 4.

Copyright © 2012–2019, Texas Instruments Incorporated

Data written to SPIDAT or SPITXBUF is transmitted to the network when appropriate edges of the SPICLK signal are received from the network master. Data written to the SPITXBUF register will be transferred to the SPIDAT register when all bits of the character to be transmitted have been shifted out of SPIDAT. If no character is currently being transmitted when SPITXBUF is written to, the data will be transferred immediately to SPIDAT. To receive data, the SPI waits for the network master to send the SPICLK signal and then shifts the data on the SPISIMO pin into SPIDAT. If data is to be transmitted by the slave simultaneously, and SPITXBUF has not been previously loaded, the data must be written to SPITXBUF or SPIDAT before the beginning of the SPICLK signal.

When the TALK bit (SPICTL.1) is cleared, data transmission is disabled, and the output line (SPISOMI) is put into the high-impedance state. If this occurs while a transmission is active, the current character is completely transmitted even though SPISOMI is forced into the high-impedance state. This ensures that the SPI is still able to receive incoming data correctly. This TALK bit allows many slave devices to be tied together on the network, but only one slave at a time is allowed to drive the SPISOMI line.

The $\overline{\text{SPISTE}}$ pin operates as the slave-select pin. An active-low signal on the $\overline{\text{SPISTE}}$ pin allows the slave SPI to transfer data to the serial data line; an inactive- high signal causes the slave SPI serial shift register to stop and its serial output pin to be put into the high-impedance state. This allows many slave devices to be tied together on the network, although only one slave device is selected at a time.

12.1.5 SPI Interrupts

This section includes information on the control bits that initialize interrupts, data format, clocking, initialization, and data transfer.

12.1.5.1 SPI Interrupt Control Bits

Five control bits are used to initialize the SPI interrupts:

- SPI INT ENA bit (SPICTL.0)
- SPI INT FLAG bit (SPISTS.6)
- OVERRUN INT ENA bit (SPICTL.4)
- RECEIVER OVERRUN FLAG bit (SPISTS.7)

12.1.5.1.1 SPI INT ENA Bit (SPICTL.0)

When the SPI interrupt-enable bit is set and an interrupt condition occurs, the corresponding interrupt is asserted.

0 Disable SPI interrupts
1 Enable SPI interrupts

12.1.5.1.2 SPI INT FLAG Bit (SPISTS.6)

This status flag indicates that a character has been placed in the SPI receiver buffer and is ready to be read.

When a complete character has been shifted into or out of SPIDAT, the SPI INT FLAG bit (SPISTS.6) is set, and an interrupt is generated if enabled by the SPI INT ENA bit (SPICTL.0). The interrupt flag remains set until it is cleared by one of the following events:

- The interrupt is acknowledged.
- The CPU reads the SPIRXBUF (reading the SPIRXEMU does not clear the SPI INT FLAG bit).
- The device enters IDLE2 or HALT mode with an IDLE instruction.
- Software clears the SPI SW RESET bit (SPICCR.7).
- A system reset occurs.

When the SPI INT FLAG bit is set, a character has been placed into the SPIRXBUF and is ready to be read. If the CPU does not read the character by the time the next complete character has been received, the new character is written into SPIRXBUF, and the RECEIVER OVERRUN Flag bit (SPISTS.7) is set.

Copyright © 2012–2019, Texas Instruments Incorporated

12.1.5.1.3 OVERRUN INT ENA Bit (SPICTL.4)

Setting the overrun interrupt enable bit allows the assertion of an interrupt whenever the RECEIVER OVERRUN Flag bit (SPISTS.7) is set by hardware. Interrupts generated by SPISTS.7 and by the SPI INT FLAG bit (SPISTS.6) share the same interrupt vector.

0 Disable RECEIVER OVERRUN Flag bit interrupts
1 Enable RECEIVER OVERRUN Flag bit interrupts

12.1.5.1.4 RECEIVER OVERRUN FLAG Bit (SPISTS.7)

The RECEIVER OVERRUN Flag bit is set whenever a new character is received and loaded into the SPIRXBUF before the previously received character has been read from the SPIRXBUF. The RECEIVER OVERRUN Flag bit must be cleared by software.

12.1.5.2 Data Format

Four bits (SPICCR.3–0) specify the number of bits (1 to 16) in the data character. This information directs the state control logic to count the number of bits received or transmitted to determine when a complete character has been processed. The following statements apply to characters with fewer than 16 bits:

- Data must be left-justified when written to SPIDAT and SPITXBUF.
- Data read back from SPIRXBUF is right-justified.
- SPIRXBUF contains the most recently received character, right-justified, plus any bits that remain from previous transmission(s) that have been shifted to the left (shown in Example 12-1).

Example 12-1. Transmission of Bit From SPIRXBUF

Conditions:

1. Transmission character length = 1 bit (specified in bits SPICCR.3−0)
2. The current value of SPIDAT = 737Bh

SPIDAT (before transmission)																
	0	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1
SPIDAT (after transmission)																
(TXed) 0 ←	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	x[(1)] ← (RXed)
SPIRXBUF (after transmission)																
	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	x[(1)]

(1) x = 1 if SPISOMI data is high; x = 0 if SPISOMI data is low; master mode is assumed.

Copyright © 2012–2019, Texas Instruments Incorporated

12.1.5.3 Baud Rate and Clocking Schemes

The SPI module supports 125 different baud rates and four different clock schemes. Depending on whether the SPI clock is in slave or master mode, the SPICLK pin can receive an external SPI clock signal or provide the SPI clock signal, respectively.

- In the slave mode, the SPI clock is received on the SPICLK pin from the external source, and can be no greater than the LSPCLK frequency divided by 4.
- In the master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin, and can be no greater than the LSPCLK frequency divided by 4.

Example 12-2 shows how to determine the SPI baud rates.

Example 12-2. Baud Rate Determination

For SPIBRR = 3 to 127:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)} \quad (5)$$

For SPIBRR = 0, 1, or 2:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4} \quad (6)$$

where:

LSPCLK = Low-speed peripheral clock frequency of the device

SPIBRR = Contents of the SPIBRR in the master SPI device

To determine what value to load into SPIBRR, you must know the device system clock (LSPCLK) frequency (which is device-specific) and the baud rate at which you will be operating.

Example 1−2 shows how to determine the maximum baud rate at which the SPI can communicate. Assume that LSPCLK = 40 MHz.

Example 12-3. Maximum Baud-Rate Calculation

$$\text{Maximum SPI Baud Rate} = \frac{\text{LSPCLK}}{4} = \frac{40 \times 10^6}{4} = 10 \times 10^6 \text{ bps} \quad (7)$$

12.1.5.3.1 SPI Clocking Schemes

The CLOCK POLARITY bit (SPICCR.6) and the CLOCK PHASE bit (SPICTL.3) control four different clocking schemes on the SPICLK pin. The CLOCK POLARITY bit selects the active edge, either rising or falling, of the clock. The CLOCK PHASE bit selects a half-cycle delay of the clock. The four different clocking schemes are as follows:

- Falling Edge Without Delay. The SPI transmits data on the falling edge of the SPICLK and receives data on the rising edge of the SPICLK.
- Falling Edge With Delay. The SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
- Rising Edge Without Delay. The SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
- Rising Edge With Delay. The SPI transmits data one half-cycle ahead of the rising edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.

Copyright © 2012–2019, Texas Instruments Incorporated

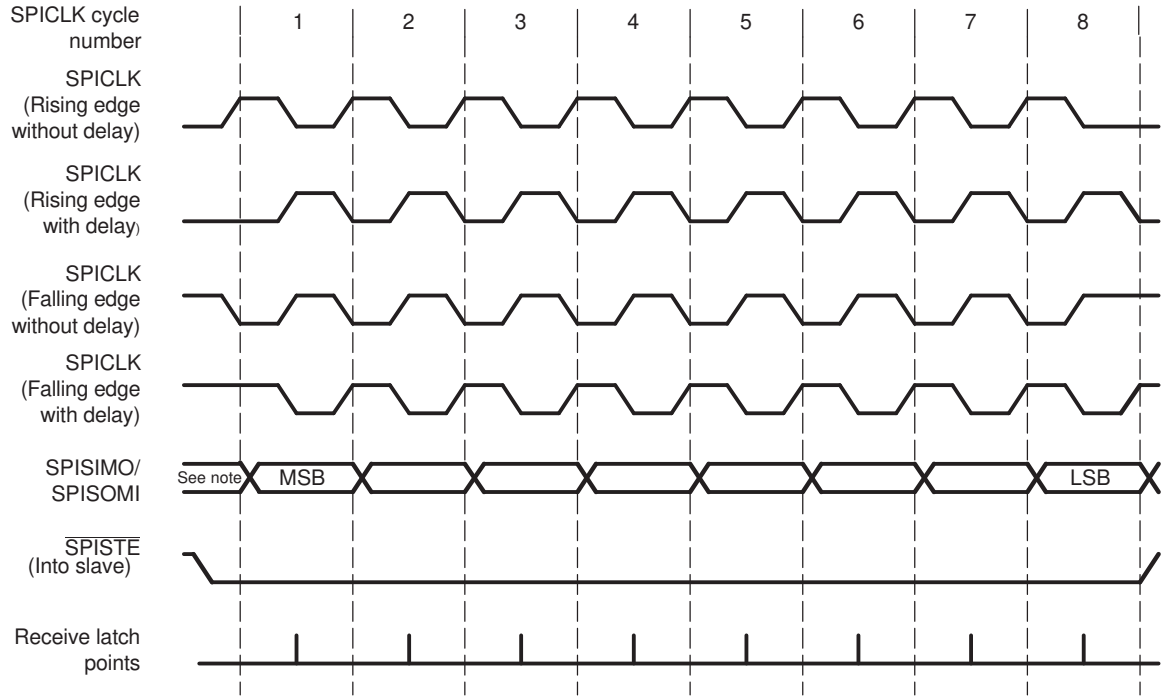
The selection procedure for the SPI clocking scheme is shown in Table 12-3. Examples of these four clocking schemes relative to transmitted and received data are shown in Figure 12-4.

Table 12-3. SPI Clocking Scheme Selection Guide

SPICLK Scheme	CLOCK POLARITY (SPICCR.6)	CLOCK PHASE (SPICTL.3) [1]
Rising edge without delay	0	0
Rising edge with delay	0	1
Falling edge without delay	1	0
Falling edge with delay	1	1

(1) The description of CLOCK PHASE and CLOCK POLARITY differs between manufacturers. For proper operation, select the desired waveform to determine the PHASE and POLARITY settings.

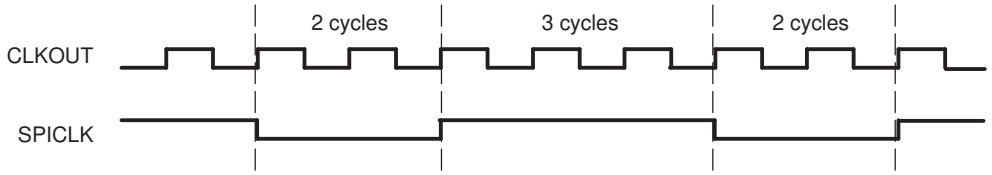
Figure 12-4. SPICLK Signal Options



Note: Previous data bit

For the SPI, SPICLK symmetry is retained only when the result of (SPIBRR+1) is an even value. When (SPIBRR + 1) is an odd value and SPIBRR is greater than 3, SPICLK becomes asymmetrical. The low pulse of SPICLK is one CLKOUT longer than the high pulse when the CLOCK POLARITY bit is clear (0). When the CLOCK POLARITY bit is set to 1, the high pulse of the SPICLK is one CLKOUT longer than the low pulse, as shown in Figure 12-5.

Figure 12-5. SPI: SPICLK-CLKOUT Characteristic When (BRR + 1) is Odd, BRR > 3, and CLOCK POLARITY = 1



Copyright © 2012–2019, Texas Instruments Incorporated

12.1.5.4 Initialization Upon Reset

A system reset forces the SPI peripheral module into the following default configuration:

- Unit is configured as a slave module (MASTER/SLAVE = 0)
- Transmit capability is disabled (TALK = 0)
- Data is latched at the input on the falling edge of the SPICLK signal
- Character length is assumed to be one bit
- SPI interrupts are disabled
- Data in SPIDAT is reset to 0000h
- SPI module pin functions are selected as general-purpose inputs (this is done in I/O MUX control register B [MCRB])

To change this SPI configuration:

Step 1. Clear the SPI SW RESET bit (SPICCR.7) to 0 to force the SPI to the reset state.

Step 2. Initialize the SPI configuration, format, baud rate, and pin functions as desired.

Step 3. Set the SPI SW RESET bit to 1 to release the SPI from the reset state.

Step 4. Write to SPIDAT or SPITXBUF (this initiates the communication process in the master).

Step 5. Read SPIRXBUF after the data transmission has completed (SPISTS.6 = 1) to determine what data was received.

To prevent unwanted and unforeseen events from occurring during or as a result of initialization changes, clear the SPI SW RESET bit (SPICCR.7) before making initialization changes, and then set this bit after initialization is complete.

NOTE: Do not change the SPI configuration when communication is in progress.

12.1.5.5 Data Transfer Example

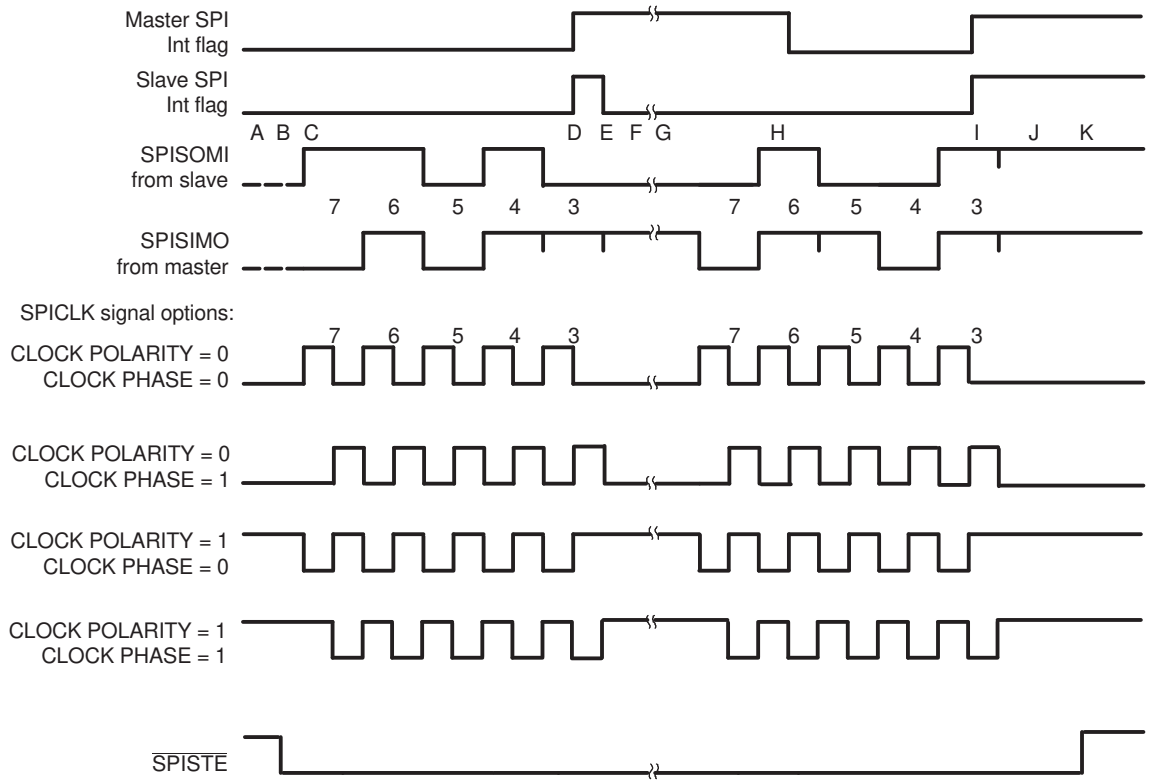
The timing diagram shown in Figure 12-6 illustrates an SPI data transfer between two devices using a character length of five bits with the SPICLK being symmetrical.

The timing diagram with SPICLK unsymmetrical (Figure 12-5) shares similar characterizations with Figure 12-6 except that the data transfer is one CLKOUT cycle longer per bit during the low pulse (CLOCK POLARITY = 0) or during the high pulse (CLOCK POLARITY = 1) of the SPICLK.

Figure 12-6 is applicable for 8-bit word length transmissions. The figure is shown for illustrative purposes only.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-6. Five Bits per Character



A Slave writes 0D0h to SPIDAT and waits for the master to shift out the data.

B Master sets the slave $\overline{\text{SPISTE}}$ signal low (active).

C Master writes 058h to SPIDAT, which starts the transmission procedure.

D First byte is finished and sets the interrupt flags.

E Slave reads 0Bh from its SPIRXBUF (right-justified).

F Slave writes 04Ch to SPIDAT and waits for the master to shift out the data.

G Master writes 06Ch to SPIDAT, which starts the transmission procedure.

H Master reads 01Ah from the SPIRXBUF (right−justified).

I Second byte is finished and sets the interrupt flags.

J Master reads 89h and the slave reads 8Dh from their respective SPIRXBUF. After the user's software masks off the unused bits, the master receives 09h and the slave receives 0Dh.

K Master clears the slave $\overline{\text{SPISTE}}$ signal high (inactive).

12.1.6 SPI FIFO Description

The following steps explain the FIFO features and help with programming the SPI FIFOs:

1. Reset. At reset the SPI powers up in standard SPI mode, the FIFO function is disabled. The FIFO registers SPIFFTX, SPIFFRX and SPIFFCT remain inactive.
2. Standard SPI. The standard SPI mode will work with SPIINT/SPIRXINT as the interrupt source.
3. Mode change. FIFO mode is enabled by setting the SPIFFENA bit to 1 in the SPIFFTX register. SPIRST can reset the FIFO mode at any stage of its operation.
4. Active registers. All the SPI registers and SPI FIFO registers SPIFFTX, SPIFFRX, and SPIFFCT will be active.
5. Interrupts. FIFO mode has two interrupts one for transmit FIFO, SPITXINT and one for receive FIFO, SPIINT/SPIRXINT. SPIINT/SPIRXINT is the common interrupt for SPI FIFO receive, receive error and receive FIFO overflow conditions. The single SPIINT for both transmit and receive sections of the standard SPI will be disabled and this interrupt will service as SPI receive FIFO interrupt.
6. Buffers. Transmit and receive buffers are supplemented with two 4x16 FIFOs. The one-word transmit

Copyright © 2012–2019, Texas Instruments Incorporated

buffer (TXBUF) of the standard SPI functions as a transition buffer between the transmit FIFO and shift register. The one-word transmit buffer will be loaded from transmit FIFO only after the last bit of the shift register is shifted out.

7. Delayed transfer. The rate at which transmit words in the FIFO are transferred to transmit shift register is programmable. The SPIFFCT register bits (7−0) FFTXDLY7−FFTXDLY0 define the delay between the word transfer. The delay is defined in number SPI serial clock cycles. The 8-bit register could define a minimum delay of 0 serial clock cycles and a maximum of 255 serial clock cycles. With zero delay, the SPI module can transmit data in continuous mode with the FIFO words shifting out back to back. With the 255 clock delay, the SPI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 255 SPI clocks between each words. The programmable delay facilitates glueless interface to various slow SPI peripherals, such as EEPROMs, ADC, DAC etc.
8. FIFO status bits. Both transmit and receive FIFOs have status bits TXFFST or RXFFST (bits 12− 0) that define the number of words available in the FIFOs at any time. The transmit FIFO reset bit TXFIFO and receive reset bit RXFIFO will reset the FIFO pointers to zero when these bits are set to 1. The FIFOs will resume operation from start once these bits are cleared to zero.
9. Programmable interrupt levels. Both transmit and receive FIFO can generate CPU interrupts. The interrupt trigger is generated whenever the transmit FIFO status bits TXFFST (bits 12−8) match (less than or equal to) the interrupt trigger level bits TXFFIL (bits 4−0). This provides a programmable interrupt trigger for transmit and receive sections of the SPI. The default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO respectively.

12.1.6.1 SPI Interrupts

Figure 12-7. SPI FIFO Interrupt Flags and Enable Logic Generation

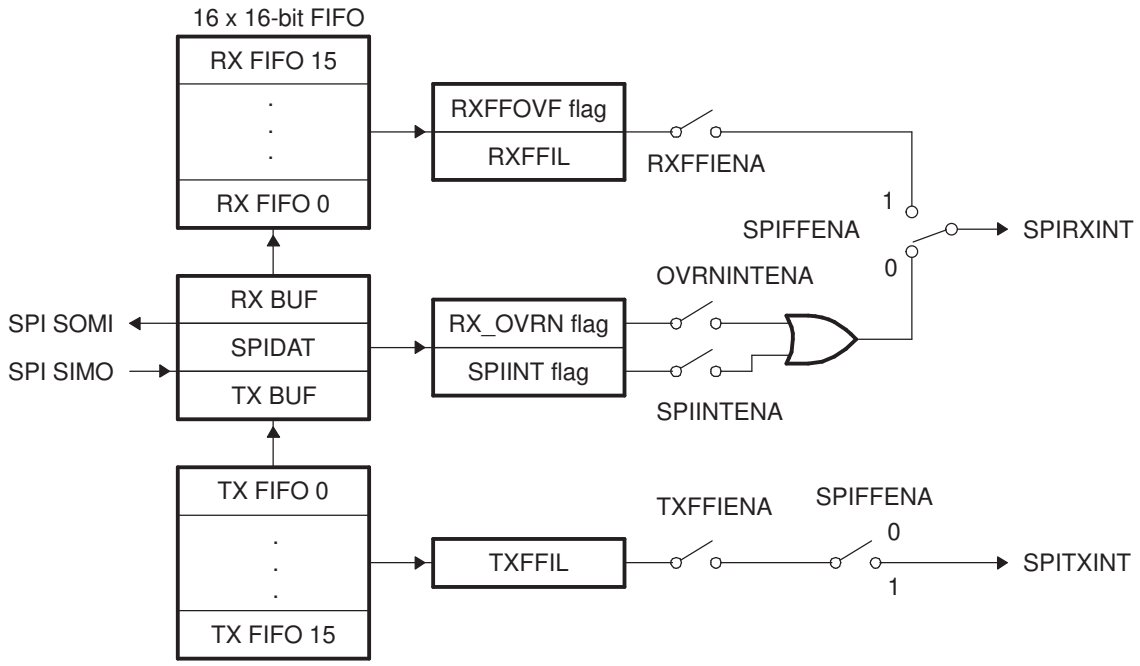


Table 12-4. SPI Interrupt Flag Modes

	SPI Interrupt	Interrupt	Interrupt	FIFO Enable	Interrupt[1]
FIFO Options	Source	Flags	Enables	SPIFFENA	line
SPI without FIFO					
	Receive overrun	RXOVRN	OVRNINTENA	0	SPIRXINT
	Data receive	SPIINT	SPIINTENA	0	SPIRXINT
	Transmit empty	SPIINT	SPIINTENA	0	SPIRXINT

(1) In non FIFO mode, SPIRXINT is the same as the SPIINT interrupt.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 12-4. SPI Interrupt Flag Modes (continued)

	SPI Interrupt	Interrupt	Interrupt	FIFO Enable	Interrupt[1]
SPI FIFO mode					
	FIFO receive	RXFFIL	RXFFIENA	1	SPIRXINT
	Transmit empty	TXFFIL	TXFFIENA	1	SPITXINT

12.1.7 SPI 3-Wire Mode Description

SPI 3-wire mode allows for SPI communication over 3 pins instead of the normal 4 pins.

In master mode, if the TRIWIRE (SPIPRI.0) bit is set, enabling 3-wire SPI mode, SPISIMOx becomes the bi-directional SPIMOMIx (SPI master out, master in) pin, and SPISOMIx is no longer used by the SPI. In slave mode, if the TRIWIRE bit is set, SPISOMIx becomes the bi-directional SPISISOx (SPI slave in, slave out) pin, and SPISIMOx is no longer used by the SPI.

The table below indicates the pin function differences between 3-wire and 4-wire SPI mode for a master and slave SPI.

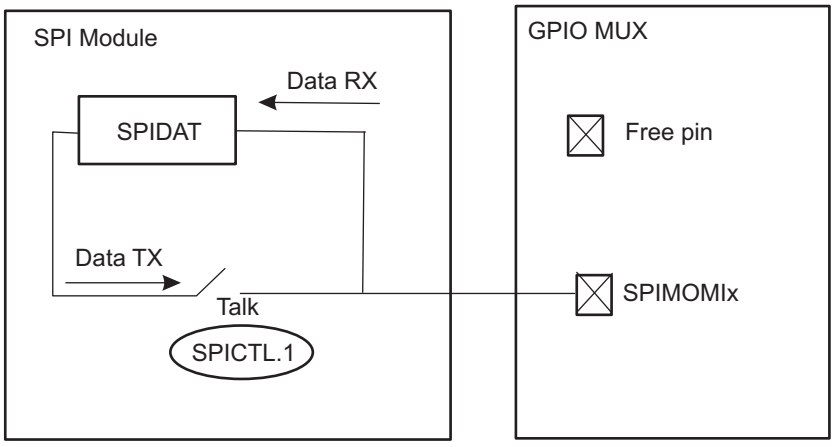
Table 12-5. 4-wire vs. 3-wire SPI Pin Functions

4-wire SPI	3-wire SPI (Master)	3-wire SPI(Slave)
SPICLKx	SPICLKx	SPICLKx
SPISTEx	SPISTEx	SPISTEx
SPISIMOx	SPIMOMIx	Free
SPISOMIx	Free	SPISISOx

Because in 3-wire mode, the receive and transmit paths within the SPI are connected, any data transmitted by the SPI module is also received by itself. The application software must take care to perform a dummy read to clear the SPI data register of the additional received data.

The TALK bit (SPICTL.1) plays an important role in 3-wire SPI mode. The bit must be set to transmit data and cleared prior to reading data. In master mode, in order to initiate a read, the application software must write dummy data to the SPI data register (SPIDAT or SPIRXBUF) while the TALK bit is cleared (no data is transmitted out the SPIMOMI pin) before reading from the data register.

Figure 12-8. SPI 3-wire Master Mode



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-9. SPI 3-wire Slave Mode

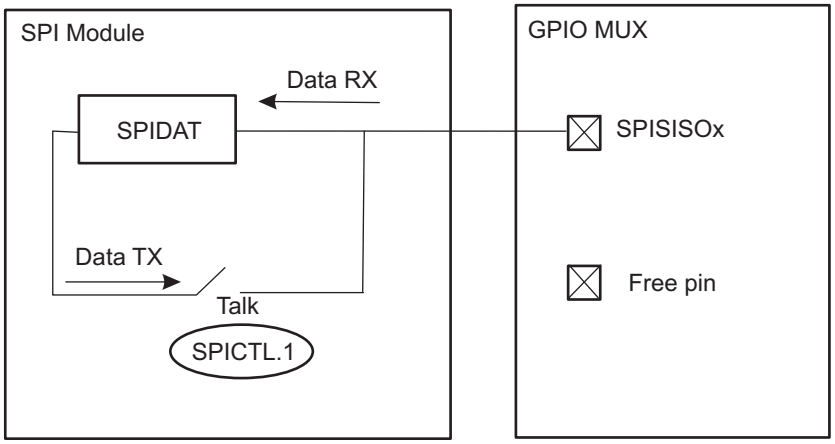


Table 12-6 indicates how data is received or transmitted in the various SPI modes while the TALK bit is set or cleared.

Table 12-6. 3-Wire SPI Pin Configuration

Pin Mode	SPIPRI[TRIWIRE]	SPICTL[TALK]	SPISIMO	SPISOMI
Master Mode				
4-wire	0	X	TX	RX
3-pin mode	1	0	RX	Disconnect from SPI
		1	TX/RX	
Slave Mode				
4-wire	0	X	RX	TX
3-pin mode	1	0	Disconnect from SPI	RX
		1		TX/RX

SPI 3-Wire Mode Code Examples

In addition to the normal SPI initialization, to configure the SPI module for 3-wire mode, the TRIWIRE bit (SPIPRI.0) must be set to 1. After initialization, there are several considerations to take into account when transmitting and receiving data in 3-wire master and slave mode. The following examples demonstrate these considerations.

In 3-wire master mode, SPICLKx, SPISTEx, and SPISIMOx pins must be configured as SPI pins (SPISOMIx pin can be configured as non-SPI pin). When the master transmits, it receives the data it transmits (because SPISIMOx and SPISOMIx are connected internally in 3-wire mode). Therefore, the junk data received must be cleared from the receive buffer every time data is transmitted.

Example 12-4. 3-Wire Master Mode Transmit

```
Uint16 data;
Uint16 dummy;

     SpiaRegs.SPICTL.bit.TALK = 1;             // Enable Transmit path
     SpiaRegs.SPITXBUF = data; // Master transmits data
     while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Waits until data rx'd
     dummy = SpiaRegs.SPIRXBUF;                // Clears junk data from itself
                                               // bc it rx'd same data tx'd
```


Copyright © 2012–2019, Texas Instruments Incorporated

To receive data in 3-wire master mode, the master must clear the TALK (SPICTL.1) bit to 0 to close the transmit path and then transmit dummy data in order to initiate the transfer from the slave. Because the TALK bit is 0, unlike in transmit mode, the master dummy data does not appear on the SPISIMOx pin, and the master does not receive its own dummy data. Instead, the data from the slave is received by the master.

Example 12-5. 3-Wire Master Mode Receive

```
Uint16 rdata;
Uint16 dummy;

     SpiaRegs.SPICTL.bit.TALK = 0;             // Disable Transmit path
     SpiaRegs.SPITXBUF = dummy;                // Send dummy to start tx
     // NOTE: because TALK = 0, data does not tx onto SPISIMOA pin
     while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Wait until data received
     rdata = SpiaRegs.SPIRXBUF;                // Master reads data
```

In 3-wire slave mode, SPICLKx, SPISTEx, and SPISOMIx pins must be configured as SPI pins (SPISIMOx pin can be configured as non-SPI pin). Like in master mode, when transmitting, the slave receives the data it transmits and must clear this junk data from its receive buffer.

Example 12-6. 3-Wire Slave Mode Transmit

```
Uint16 data;
Uint16 dummy;
     SpiaRegs.SPICTL.bit.TALK = 1;             // Enable Transmit path
     SpiaRegs.SPITXBUF = data;                 // Slave transmits data
     while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Wait until data rx'd
     dummy = SpiaRegs.SPIRXBUF;                // Clears junk data from itself
```

As in 3-wire master mode, the TALK bit must be cleared to 0. Otherwise, the slave receives data normally.

Example 12-7. - 3-Wire Slave Mode Receive

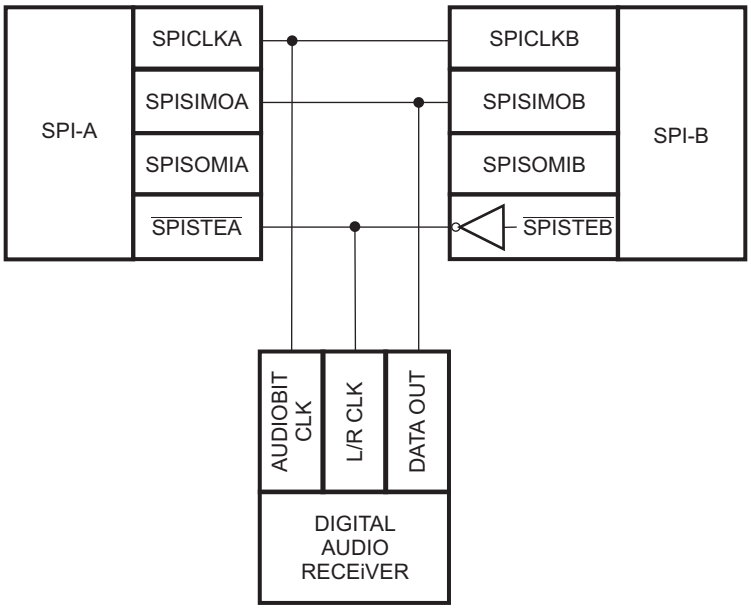
```
Uint16 rdata;
    SpiaRegs.SPICTL.bit.TALK = 0;             // Disable Transmit path
    while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Waits until data rx'd
    rdata = SpiaRegs.SPIRXBUF;                // Slave reads data
```

12.1.8 SPI STEINV Bit in Digital Audio Transfers

On those devices with 2 SPI modules, enabling the STEINV bit (SPIPRI.1) on one of the SPI modules allows the pair of SPIs to receive both left and right-channel digital audio data in slave mode. The SPI module that receives a normal active-low $\overline{\text{SPISTE}}$ signal stores right-channel data, and the SPI module that receives an inverted active-high $\overline{\text{SPISTE}}$ signal stores left-channel data from the master. To receive digital audio data from a digital audio interface receiver, the SPI modules can be connected as shown in Figure 12-10.

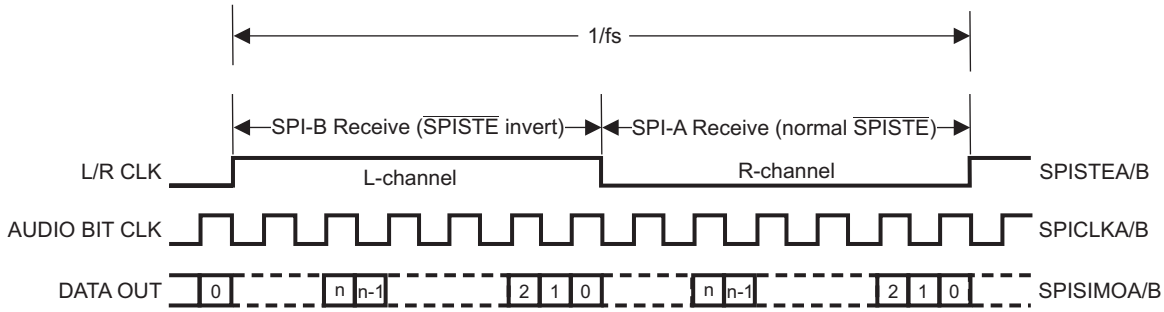
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-10. SPI Digital Audio Receiver Configuration Using 2 SPIs



Standard 28x SPI timing requirements limit the number of digital audio interface formats supported using the 2-SPI configuration with the STEINV bit. See your device-specific data sheet electricals for SPI timing requirements. With the SPI clock phase configured such that the CLOCK POLARITY (SPICCR.6) bit is 0 and the CLOCK PHASE (SPICTL.3) bit is 1 (data latched on rising edge of clock), standard right-justified digital audio interface data format is supported as shown in Figure 12-11.

Figure 12-11. Standard Right-Justified Digital Audio Data Format



12.2 C28 SPI-A to M3 SSI3 Internal Loopback

The C28 SPI-A peripheral can be internally connected to the M3 SSI peripheral. External GPIO pins are not used when the loopback feature is enabled and can be used for other functions.

Figure 12-12 illustrates the loopback connections between the M3 SSI3 and C28 SPI-A. The internal connection logic block handles the signal routing between the peripherals and the GPIO mux.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-12. SSI and SPI Connections for Loopback Mode

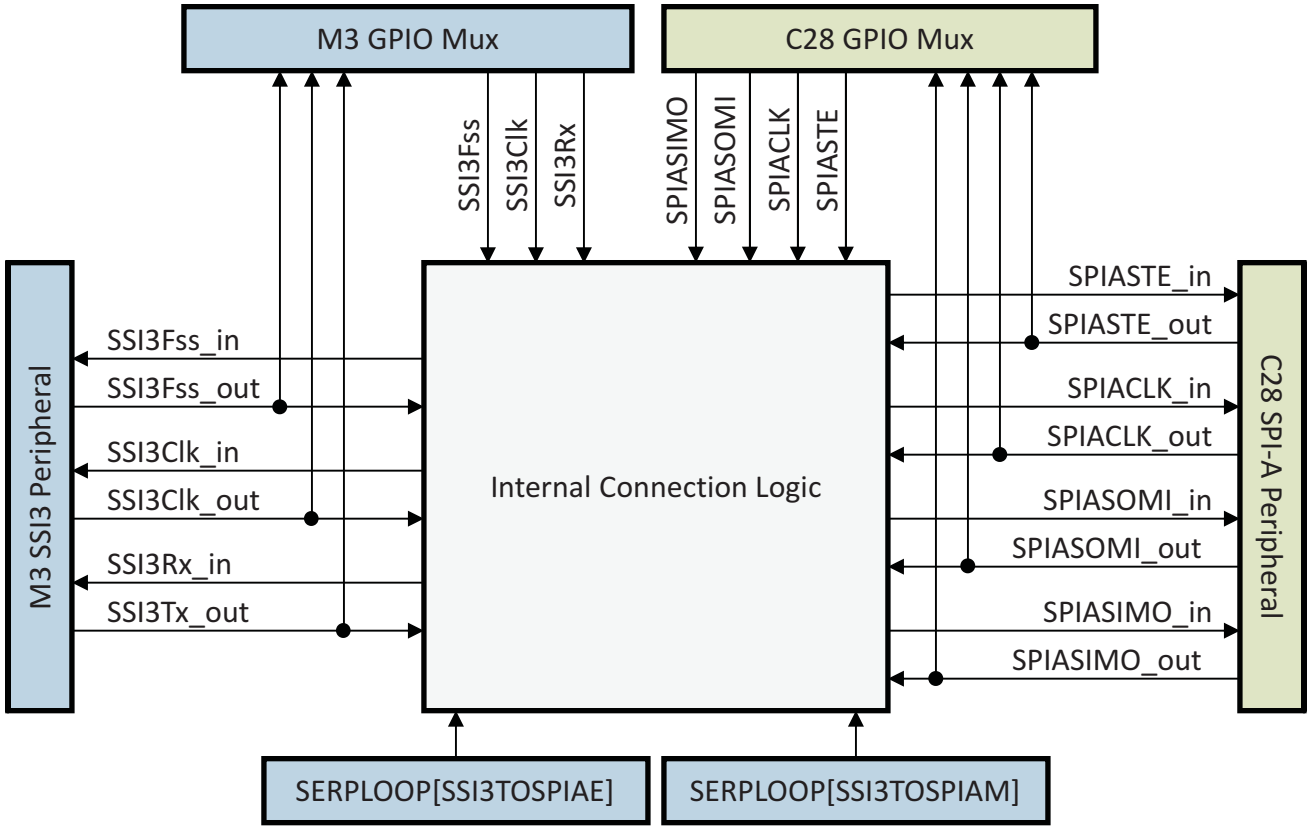


Table 12-7 shows the three loopback modes available. Loopback mode is disabled by default. The SERPLOOP register in the M3 system control register space determines the loopback mode. Configuration of SERPLOOP is only available on the M3 master subsystem. Three different modes are available: not connected (default), M3 SSI3 master connected to C28 SPI-A slave, and M3 SSI3 slave connected to C28 SPI-A master. For a detailed description of the SERPLOOP register, refer to the *System control and Interrupts* chapter.

Table 12-7. Loopback Modes

SERPLOOP[SSI3TOSPIAE]	SERPLOOP[SSI3TOSPIAM]	Mode
0	x	Not Connected (default on reset)
1	0	M3 SSI3 Master connected to C28 SPI-A Slave
1	1	M3 SSI3 Slave connected to C28 SPI-A Master
Legend: 0 = Bit cleared, 1 = Bit set, X = Any		

12.2.1 Loopback Initialization and Configuration

To enable M3 SSI3 master to C28 SPI-A slave loopback mode, perform the following steps:

1. Enable and configure the M3 SSI module as a SPI master by following the steps outlined in the Initialization and Configuration section of the *M3 Synchronous Serial Interface (SSI)* chapter.
2. Enable the corresponding loopback mode by setting SERPLOOP[SSI3TOSPIA] = 0x2 on the M3 subsystem.
3. Enable and configure the C28 SPI module as detailed in Section 12.1.5.4.

To enable M3 SSI3 slave to C28 SPI-A master loopback mode, perform the following steps:

Copyright © 2012–2019, Texas Instruments Incorporated

1. Enable and configure the M3 SSI module as a SPI slave by following the steps outlined in the Initialization and Configuration section of the *M3 Synchronous Serial Interface (SSI)* chapter.
2. Enable the corresponding loopback mode by setting SERPLOOP[SSI3TOSPIA] = 0x3 on the M3 subsystem.
3. Enable and configure the C28 SPI module as detailed in Section 12.1.5.4.

To disable loopback between the M3 and C28, set SERPLOOP[SSI3TOSPIA] = 0 on the M3 subsystem.

NOTE: Ensure that compatible CPOL and CPHA modes are used. The definition of the M3 CPOL and CPHA bits are different from the C28 CPOL and CPHA bits. Using incompatible configurations can cause data to be shifted by one bit.

Copyright © 2012–2019, Texas Instruments Incorporated

12.3 SPI Registers and Waveforms

This section contains the registers, bit descriptions, and waveforms.

12.3.1 SPI Control Registers

The SPI is controlled and accessed through registers in the control register file.

12.3.1.1 SPI Configuration Control Register (SPICCR)

SPICCR controls the setup of the SPI for operation.

Figure 12-13. SPI Configuration Control Register (SPICCR) — Address 7040h

7	6	5	4	3	2	1	0
SPI SW Reset	CLOCK POLARITY	Reserved	SPILBK	SPI CHAR3	SPI CHAR2	SPI CHAR1	SPI CHAR0
R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-8. SPI Configuration Control Register (SPICCR) Field Descriptions

Bit	Field	Value	Description
7	SPI SW RESET		SPI software reset. When changing configuration, you should clear this bit before making changes and set this bit before resuming operation.
		0	Initializes the SPI operating flags to the reset condition. Specifically, the RECEIVER OVERRUN Flag bit (SPISTS.7), the SPI INT FLAG bit (SPISTS.6), and the TXBUF FULL Flag bit (SPISTS.5) are cleared. SPISTE will become inactive. SPICLK will be immediately driven to 0 regardless of the clock polarity. The SPI configuration remains unchanged.
		1	SPI is ready to transmit or receive the next character. When the SPI SW RESET bit is a 0, a character written to the transmitter will not be shifted out when this bit is set. A new character must be written to the serial data register. SPICLK will be returned to its inactive state one SPICLK cycle after this bit is set.
6	CLOCK POLARITY		Shift Clock Polarity. This bit controls the polarity of the SPICLK signal. CLOCK POLARITY and CLOCK PHASE (SPICTL.3) control four clocking schemes on the SPICLK pin. See Section 12.1.5.3.
		0	Data is output on rising edge and input on falling edge. When no SPI data is sent, SPICLK is at low level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows: • CLOCK PHASE = 0: Data is output on the rising edge of the SPICLK signal; input data is latched on the falling edge of the SPICLK signal. • CLOCK PHASE = 1: Data is output one half-cycle before the first rising edge of the SPICLK signal and on subsequent falling edges of the SPICLK signal; input data is latched on the rising edge of the SPICLK signal.
		1	Data is output on falling edge and input on rising edge. When no SPI data is sent, SPICLK is at high level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows: • CLOCK PHASE = 0: Data is output on the falling edge of the SPICLK signal; input data is latched on the rising edge of the SPICLK signal. • CLOCK PHASE = 1: Data is output one half-cycle before the first falling edge of the SPICLK signal and on subsequent rising edges of the SPICLK signal; input data is latched on the falling edge of the SPICLK signal.
5	Reserved		Reads return zero; writes have no effect.
4	SPILBK		SPI loopback. Loop back mode allows module validation during device testing. This mode is valid only in master mode of the SPI.
		0	SPI loop back mode disabled – default value after reset
		1	SPI loop back mode enabled, SIMO/SOMI lines are connected internally. Used for module self tests.
3-0	SPI CHAR3 − SPI CHAR0		Character Length Control Bits 3-0. These four bits determine the number of bits to be shifted in or out as a single character during one shift sequence. Table 12-9 lists the character length selected by the bit values.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 12-9. Character Length Control Bit Values

SPI CHAR3	SPI CHAR2	SPI CHAR1	SPI CHAR0	Character Length
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10
1	0	1	0	11
1	0	1	1	12
1	1	0	0	13
1	1	0	1	14
1	1	1	0	15
1	1	1	1	16

12.3.1.2 SPI Operation Control Register (SPICTL)

SPICTL controls data transmission, the SPI's ability to generate interrupts, the SPICLK phase, and the operational mode (slave or master).

Figure 12-14. SPI Operation Control Register (SPICTL) — Address 7041h

7	6	5	4	3	2	1	0
Reserved			OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-10. SPI Operation Control Register (SPICTL) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved		Reads return zero; writes have no effect.
4	Overrun INT ENA		Overrun Interrupt Enable. Setting this bit causes an interrupt to be generated when the RECEIVER OVERRUN Flag bit (SPISTS.7) is set by hardware. Interrupts generated by the RECEIVER OVERRUN Flag bit and the SPI INT FLAG bit (SPISTS.6) share the same interrupt vector.
		0	Disable RECEIVER OVERRUN Flag bit (SPISTS.7) interrupts
		1	Enable RECEIVER OVERRUN Flag bit (SPISTS.7) interrupts
3	CLOCK PHASE		SPI Clock Phase Select. This bit controls the phase of the SPICLK signal. CLOCK PHASE and CLOCK POLARITY (SPICCR.6) make four different clocking schemes possible (see Figure 12-4). When operating with CLOCK PHASE high, the SPI (master or slave) makes the first bit of data available after SPIDAT is written and before the first edge of the SPICLK signal, regardless of which SPI mode is being used.
		0	Normal SPI clocking scheme, depending on the CLOCK POLARITY bit (SPICCR.6)
		1	SPICLK signal delayed by one half-cycle; polarity determined by the CLOCK POLARITY bit
2	MASTER / SLAVE		SPI Network Mode Control. This bit determines whether the SPI is a network master or slave. During reset initialization, the SPI is automatically configured as a network slave.
		0	SPI configured as a slave.
		1	SPI configured as a master.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 12-10. SPI Operation Control Register (SPICTL) Field Descriptions (continued)

Bit	Field	Value	Description
1	TALK		Master/Slave Transmit Enable. The TALK bit can disable data transmission (master or slave) by placing the serial data output in the high-impedance state. If this bit is disabled during a transmission, the transmit shift register continues to operate until the previous character is shifted out. When the TALK bit is disabled, the SPI is still able to receive characters and update the status flags. TALK is cleared (disabled) by a system reset.
		0	Disables transmission: • Slave mode operation: If not previously configured as a general-purpose I/O pin, the SPISOMI pin will be put in the high-impedance state. • Master mode operation: If not previously configured as a general-purpose I/O pin, the SPISIMO pin will be put in the high-impedance state.
		1	Enables transmission For the 4-pin option, ensure to enable the receiver's $\overline{\text{SPISTE}}$ input pin.
0	SPI INT ENA		SPI Interrupt Enable. This bit controls the SPI's ability to generate a transmit/receive interrupt. The SPI INT FLAG bit (SPISTS.6) is unaffected by this bit.
		0	Disables interrupt
		1	Enables interrupt

12.3.1.3 SPI Status Register (SPIST)

Figure 12-15. SPI Status Register (SPIST) — Address 7042h

7	6	5	4 ... 0
RECEIVER OVERRUN FLAG[1] [2]	SPI INT FLAG[1] [2]	TX BUF FULL FLAG[2]	Reserved
R/C-0	R/C-0	R/C-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

(1) The RECEIVER OVERRUN FLAG bit and the SPI INT FLAG bit share the same interrupt vector.
(2) Writing a 0 to bits 5, 6, and 7 has no effect.

Table 12-11. SPI Status Register (SPIST) Field Descriptions

Bit	Field	Value	Description
7	RECEIVER OVERRUN FLAG		SPI Receiver Overrun Flag. This bit is a read/clear-only flag. The SPI hardware sets this bit when a receive or transmit operation completes before the previous character has been read from the buffer. The bit indicates that the last received character has been overwritten and therefore lost (when the SPIRXBUF was overwritten by the SPI module before the previous character was read by the user application). The SPI requests one interrupt sequence each time this bit is set if the OVERRUN INT ENA bit (SPICTL.4) is set high. The bit is cleared in one of three ways: • Writing a 1 to this bit • Writing a 0 to SPI SW RESET (SPICCR.7) • Resetting the system If the OVERRUN INT ENA bit (SPICTL.4) is set, the SPI requests only one interrupt upon the first occurrence of setting the RECEIVER OVERRUN Flag bit. Subsequent overruns will not request additional interrupts if this flag bit is already set. This means that in order to allow new overrun interrupt requests the user must clear this flag bit by writing a 1 to SPISTS.7 each time an overrun condition occurs. In other words, if the RECEIVER OVERRUN Flag bit is left set (not cleared) by the interrupt service routine, another overrun interrupt will not be immediately re-entered when the interrupt service routine is exited.
		0	Writing a 0 has no effect
		1	Clears this bit. The RECEIVER OVERRUN Flag bit should be cleared during the interrupt service routine because the RECEIVER OVERRUN Flag bit and SPI INT FLAG bit (SPISTS.6) share the same interrupt vector. This will alleviate any possible doubt as to the source of the interrupt when the next byte is received.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 12-11. SPI Status Register (SPIST) Field Descriptions (continued)

Bit	Field	Value	Description
6	SPI INT FLAG		SPI Interrupt Flag. SPI INT FLAG is a read-only flag. The SPI hardware sets this bit to indicate that it has completed sending or receiving the last bit and is ready to be serviced. The received character is placed in the receiver buffer at the same time this bit is set. This flag causes an interrupt to be requested if the SPI INT ENA bit (SPICTL.0) is set.
		0	Writing a 0 has no effect
		1	This bit is cleared in one of three ways: • Reading SPIRXBUF • Writing a 0 to SPI SW RESET (SPICCR.7) • Resetting the system
5	TX BUF FULL FLAG		SPI Transmit Buffer Full Flag. This read-only bit gets set to 1 when a character is written to the SPI Transmit buffer SPITXBUF. It is cleared when the character is automatically loaded into SPIDAT when the shifting out of a previous character is complete.
		0	Writing a 0 has no effect
		1	This bit is cleared at reset.
4-0	Reserved	0	Reads return zero; writes have no effect.

12.3.1.4 SPI Baud Rate Register (SPIBRR)

SPIBRR contains the bits used for baud-rate selection.

Figure 12-16. SPI Baud Rate Register (SPIBRR) — Address 7044h

7	6	5	4	3	2	1	0
Reserved	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-12. Field Descriptions

Bit	Field	Value	Description
7	Reserved		Reads return zero; writes have no effect.
6-0	SPI BIT RATE 6− SPI BIT RATE 0		SPI Bit Rate (Baud) Control. These bits determine the bit transfer rate if the SPI is the network master. There are 125 data-transfer rates (each a function of the CPU clock, LSPCLK) that can be selected. One data bit is shifted per SPICLK cycle. (SPICLK is the baud rate clock output on the SPICLK pin.) If the SPI is a network slave, the module receives a clock on the SPICLK pin from the network master; therefore, these bits have no effect on the SPICLK signal. The frequency of the input clock from the master should not exceed the slave SPI's LSPCLK signal divided by 4. In master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin. The SPI baud rates are determined by the following formula: For SPIBRR = 3 to 127: $\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)}$ For SPIBRR = 0, 1, or 2: $\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4}$ where: LSPCLK = Function of CPU clock frequency X low-speed peripheral clock of the device SPIBRR = Contents of the SPIBRR in the master SPI device

12.3.1.5 SPI Emulation Buffer Register (SPIRXEMU)

SPIRXEMU contains the received data. Reading SPIRXEMU does not clear the SPI INT FLAG bit (SPISTS.6). This is not a real register but a dummy address from which the contents of SPIRXBUF can be read by the emulator without clearing the SPI INT FLAG.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-17. SPI Emulation Buffer Register (SPIRXEMU) — Address 7046h

15	14	13	12	11	10	9	8
ERXB15	ERXB14	ERXB13	ERXB12	ERXB11	ERXB10	ERXB9	ERXB8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ERXB7	ERXB6	ERXB5	ERXB4	ERXB3	ERXB2	ERXB1	ERXB0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-13. SPI Emulation Buffer Register (SPIRXEMU) Field Descriptions

Bit	Field	Value	Description
15-0	ERXB15− ERXB0		Emulation Buffer Received Data. SPIRXEMU functions almost identically to SPIRXBUF, except that reading SPIRXEMU does not clear the SPI INT FLAG bit (SPISTS.6). Once the SPIDAT has received the complete character, the character is transferred to SPIRXEMU and SPIRXBUF, where it can be read. At the same time, SPI INT FLAG is set. This mirror register was created to support emulation. Reading SPIRXBUF clears the SPI INT FLAG bit (SPISTS.6). In the normal operation of the emulator, the control registers are read to continually update the contents of these registers on the display screen. SPIRXEMU was created so that the emulator can read this register and properly update the contents on the display screen. Reading SPIRXEMU does not clear the SPI INT FLAG bit, but reading SPIRXBUF clears this flag. In other words, SPIRXEMU enables the emulator to emulate the true operation of the SPI more accurately. It is recommended that you view SPIRXEMU in the normal emulator run mode.

12.3.1.6 SPI Serial Receive Buffer Register (SPIRXBUF)

SPIRXBUF contains the received data. Reading SPIRXBUF clears the SPI INT FLAG bit (SPISTS.6).

Figure 12-18. SPI Serial Receive Buffer Register (SPIRXBUF) — Address 7047h

15	14	13	12	11	10	9	8
RXB15	RXB14	RXB13	RXB12	RXB11	RXB10	RXB9	RXB8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-14. SPI Serial Receive Buffer Register (SPIRXBUF) Field Descriptions

Bit	Field	Value	Description
15-0	RXB15 − RXB0		Received Data. Once SPIDAT has received the complete character, the character is transferred to SPIRXBUF, where it can be read. At the same time, the SPI INT FLAG bit (SPISTS.6) is set. Since data is shifted into the SPI's most significant bit first, it is stored right-justified in this register.

12.3.1.7 SPI Serial Transmit Buffer Register (SPITXBUF)

SPITXBUF stores the next character to be transmitted. Writing to this register sets the TX BUF FULL Flag bit (SPISTS.5). When transmission of the current character is complete, the contents of this register are automatically loaded in SPIDAT and the TX BUF FULL Flag is cleared. If no transmission is currently active, data written to this register falls through into the SPIDAT register and the TX BUF FULL Flag is not set.

In master mode, if no transmission is currently active, writing to this register initiates a transmission in the same manner that writing to SPIDAT does.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-19. SPI Serial Transmit Buffer Register (SPITXBUF) — Address 7048h

15	14	13	12	11	10	9	8
TXB15	TXB14	TXB13	TXB12	TXB11	TXB10	TXB9	TXB8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-15. SPI Serial Transmit Buffer Register (SPITXBUF) Field Descriptions

Bit	Field	Value	Description
15-0	TXB15 − TXB0		Transmit Data Buffer. This is where the next character to be transmitted is stored. When the transmission of the current character has completed, if the TX BUF FULL Flag bit is set, the contents of this register is automatically transferred to SPIDAT, and the TX BUF FULL Flag is cleared. Writes to SPITXBUF must be left-justified.

12.3.1.8 SPI Serial Data Register (SPIDAT)

SPIDAT is the transmit/receive shift register. Data written to SPIDAT is shifted out (MSB) on subsequent SPICLK cycles. For every bit (MSB) shifted out of the SPI, a bit is shifted into the LSB end of the shift register.

Figure 12-20. SPI Serial Data Register (SPIDAT) — Address 7049h

15	14	13	12	11	10	9	8
SDAT15	SDAT14	SDAT13	SDAT12	SDAT11	SDAT10	SDAT9	SDAT8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-16. SPI Serial Data Register (SPIDAT) Field Descriptions

Bit	Field	Value	Description
15-0	SDAT15 − SDAT0		Serial data. Writing to the SPIDAT performs two functions: • It provides data to be output on the serial output pin if the TALK bit (SPICTL.1) is set. • When the SPI is operating as a master, a data transfer is initiated. When initiating a transfer, see the CLOCK POLARITY bit (SPICCR.6) described in Section 12.3.1.1 and the CLOCK PHASE bit (SPICTL.3) described in Section 12.3.1.2, for the requirements. In master mode, writing dummy data to SPIDAT initiates a receiver sequence. Since the data is not hardware-justified for characters shorter than sixteen bits, transmit data must be written in left-justified form, and received data read in right-justified form.

Copyright © 2012–2019, Texas Instruments Incorporated

12.3.1.9 SPI FIFO Transmit, Receive, and Control Registers

Figure 12-21. SPI FIFO Transmit (SPIFFTX) Register − Address 704Ah

15	14	13	12	11	10	9	8
SPIRST	SPIFFENA	TXFIFO	TXFFST4	TXFFST3	TXFFST2	TXFFST1	TXFFST0
R/W-1	R/W−0	R/W-1	R−0	R−0	R−0	R−0	R−0
7	6	5	4	3	2	1	0
TXFFINT Flag	TXFFINT CLR	TXFFIENA	TXFFIL4	TXFFIL3	TXFFIL2	TXFFIL1	TXFFIL0
R/W-0	W−0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-17. SPI FIFO Transmit (SPIFFTX) Register Field Descriptions

Bit	Field	Value	Description
15	SPIRST		SPI reset
		0	Write 0 to reset the SPI transmit and receive channels. The SPI FIFO register configuration bits will be left as is.
		1	SPI FIFO can resume transmit or receive. No effect to the SPI registers bits.
14	SPIFFENA		SPI FIFO enhancements enable
		0	SPI FIFO enhancements are disabled
		1	SPI FIFO enhancements are enabled
13	TXFIFO Reset		Transmit FIFO reset
		0	Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1	Re-enable Transmit FIFO operation
12-8	TXFFST4−0		Transmit FIFO status
		00000	Transmit FIFO is empty.
		00001	Transmit FIFO has 1 word.
		00010	Transmit FIFO has 2 words.
		00011	Transmit FIFO has 3 words.
		00100	Transmit FIFO has 4words, the maximum
7	TXFFINT		TXFIFO interrupt
		0	TXFIFO interrupt has not occurred, This is a read-only bit.
		1	TXFIFO interrupt has occurred, This is a read-only bit.
6	TXFFINT CLR		TXFIFO clear
		0	Write 0 has no effect on TXFFINT flag bit, Bit reads back a zero.
		1	Write 1 to clear TXFFINT flag in bit 7.
5	TXFFIENA		TX FIFO interrupt enable
		0	TX FIFO interrupt based on TXFFIL match (less than or equal to) will be disabled .
		1	TX FIFO interrupt based on TXFFIL match (less than or equal to) will be enabled.
4-0	TXFFIL4−0		TXFFIL4−0 transmit FIFO interrupt level bits. Transmit FIFO will generate interrupt when the FIFO status bits (TXFFST4−0) and FIFO level bits (TXFFIL4−0) match (less than or equal to).
		00000	Default value is 0x00000.

Figure 12-22. SPI FIFO Receive (SPIFFRX) Register − Address 704Bh

15	14	13	12	11	10	9	8
RXFFOVF Flag	RXFFOVF CLR	RXFIFO Reset	RXFFST4	RXFFST3	RXFFST2	RXFFST1	RXFFST0
R-0	W−0	R/W−1	R−0	R−0	R−0	R−0	R−0
7	6	5	4	3	2	1	0
RXFFINT Flag	RXFFINT CLR	RXFFIENA	RXFFIL4	RXFFIL3	RXFFIL2	RXFFIL1	RXFFIL0
R-0	W−0	R/W−0	R/W−1	R/W−1	R/W−1	R/W−1	R/W−1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 12-18. SPI FIFO Receive (SPIFFRX) Register Field Descriptions

Bit	Field	Value	Description
15	RXFFOVF		Receive FIFO overflow flag
		0	Receive FIFO has not overflowed. This is a read-only bit.
		1	Receive FIFO has overflowed, read-only bit. More than 4 words have been received in to the FIFO, and the first received word is lost.
14	RXFFOVF CLR		Receive FIFO overflow clear
		0	Write 0 does not affect RXFFOVF flag bit, Bit reads back a zero
		1	Write 1 to clear RXFFOVF flag in bit 15
13	RXFIFO Reset		Receive FIFO reset
		0	Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1	Re-enable receive FIFO operation
12-8	RXFFST4−0		Receive FIFO Status
		00000	Receive FIFO is empty.
		00001	Receive FIFO has 1 word.
		00010	Receive FIFO has 2 words.
		00011	Receive FIFO has 3 words.
		00100	Receive FIFO has 4 words, the maximum.
7	RXFFINT		Receive FIFO interrupt
		0	RXFIFO interrupt has not occurred. This is a read-only bit.
		1	RXFIFO interrupt has occurred. This is a read-only bit.
6	RXFFINT CLR		Receive FIFO interrupt clear
		0	Write 0 has no effect on RXFFINT flag bit, Bit reads back a zero.
		1	Write 1 to clear RXFFINT flag in bit 7.
5	RXFFIENA		RX FIFO interrupt enable
		0	RX FIFO interrupt based on RXFFIL match (greater than or equal to) will be disabled.
		1	RX FIFO interrupt based on RXFFIL match (greater than or equal to) will be enabled.
4-0	RXFFIL4−0		Receive FIFO interrupt level bits
		11111	Receive FIFO generates an interrupt when the FIFO status bits (RXFFST4–0) are greater than or equal to the FIFO level bits (RXFFIL4–0). The default value of these bits after reset is 11111. This avoids frequent interrupts after reset, as the receive FIFO will be empty most of the time.

Figure 12-23. SPI FIFO Control (SPIFFCT) Register − Address 704Ch

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
FFTXDLY7	FFTXDLY6	FFTXDLY5	FFTXDLY4	FFTXDLY3	FFTXDLY2	FFTXDLY1	FFTXDLY0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 12-19. SPI FIFO Control (SPIFFCT) Register Field Descriptions

Bit	Field	Value	Description
15-8	Reserved		Reads return zero; writes have no effect.
7-0	FFTXDLY7−0		FIFO transmit delay bits
		0	These bits define the delay between every transfer from FIFO transmit buffer to transmit shift register. The delay is defined in number SPI serial clock cycles. The 8-bit register could define a minimum delay of 0 serial clock cycles and a maximum of 255 serial clock cycles.
		1	In FIFO mode, the buffer (TXBUF) between the shift register and the FIFO should be filled only after the shift register has completed shifting of the last bit. This is required to pass on the delay between transfers to the data stream. In the FIFO mode TXBUF should not be treated as one additional level of buffer.

Copyright © 2012–2019, Texas Instruments Incorporated

12.3.1.10 SPI Priority Control Register (SPIPRI)

Figure 12-24. SPI Priority Control Register (SPIPRI) — Address 704Fh

15							8
Reserved							
R-0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		SPI SUSP SOFT	SPI SUSP FREE	Reserved		STEINV	TRIWIRE
R-0		R/W-0	R/W-0	R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

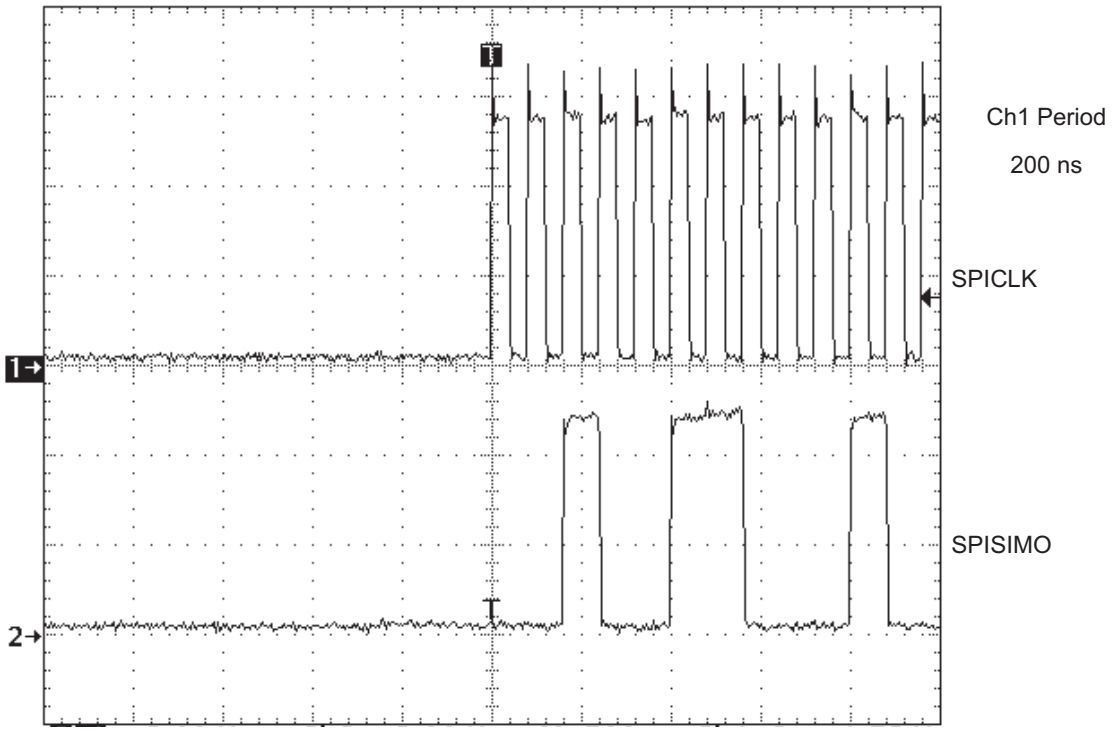
Table 12-20. SPI Priority Control Register (SPIPRI) Field Descriptions

Bit	Field	Value	Description
15-6	Reserved		Reads return zero; writes have no effect.
5-4	SPI SUSP SOFT SPI SUSP FREE		These bits determine what occurs when an emulation suspend occurs (for example, when the debugger hits a breakpoint). The peripheral can continue whatever it is doing (free-run mode) or, if in stop mode, it can either stop immediately or stop when the current operation (the current receive/transmit sequence) is complete.
		0 0	Transmission stops after midway in the bit stream while TSUSPEND is asserted. Once TSUSPEND is deasserted without a system reset, the remainder of the bits pending in the DATBUF are shifted. Example: If SPIDAT has shifted 3 out of 8 bits, the communication freezes right there. However, if TSUSPEND is later deasserted without resetting the SPI, SPI starts transmitting from where it had stopped (fourth bit in this case) and will transmit 8 bits from that point. The SCI module operates differently.
		1 0	If the emulation suspend occurs before the start of a transmission, (that is, before the first SPICLK pulse) then the transmission will not occur. If the emulation suspend occurs after the start of a transmission, then the data will be shifted out to completion. When the start of transmission occurs is dependent on the baud rate used. *Standard SPI mode:* Stop after transmitting the words in the shift register and buffer. That is, after TXBUF and SPIDAT are empty. *In FIFO mode:* Stop after transmitting the words in the shift register and buffer. That is, after TX FIFO and SPIDAT are empty.
		x 1	Free run, continue SPI operation regardless of suspend or when the suspend occurred.
3-2	Reserved		Reserved
1	STEINV		$\overline{\text{SPISTE}}$ inversion bit On devices with 2 SPI modules, inverting the $\overline{\text{SPISTE}}$ signal on one of the modules allows the device to receive left and right- channel digital audio data. This bit is only applicable to slave mode. Writing to this bit while configured as master (MASTER_SLAVE = 1) has no effect.
		0	$\overline{\text{SPISTE}}$ is active low (normal)
		1	$\overline{\text{SPISTE}}$ is active high (inverted)
0	TRIWIRE		SPI 3-wire mode enable
		0	Normal 4-wire SPI mode
		1	3-wire SPI mode enabled. The unused pin becomes a GPIO pin. In master mode, the SPISIMO pin becomes the SPIMOMI (master receive and transmit) pin and SPISOMI is free for non-SPI use. In slave mode, the SIISOMI pin becomes the SPISISO (slave receive and transmit) pin and SPISIMO is free for non-SPI use.

Copyright © 2012–2019, Texas Instruments Incorporated

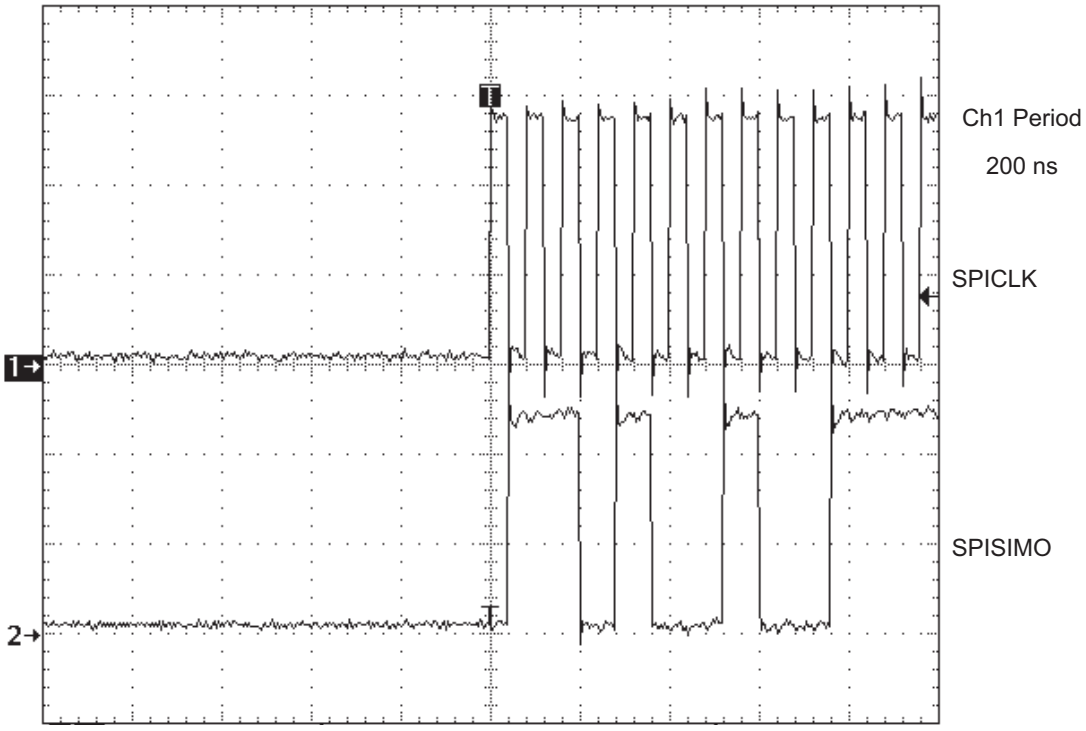
12.3.2 SPI Example Waveforms

Figure 12-25. CLOCK POLARITY = 0, CLOCK PHASE = 0 (All data transitions are during the rising edge, non-delayed clock. Inactive level is low.)



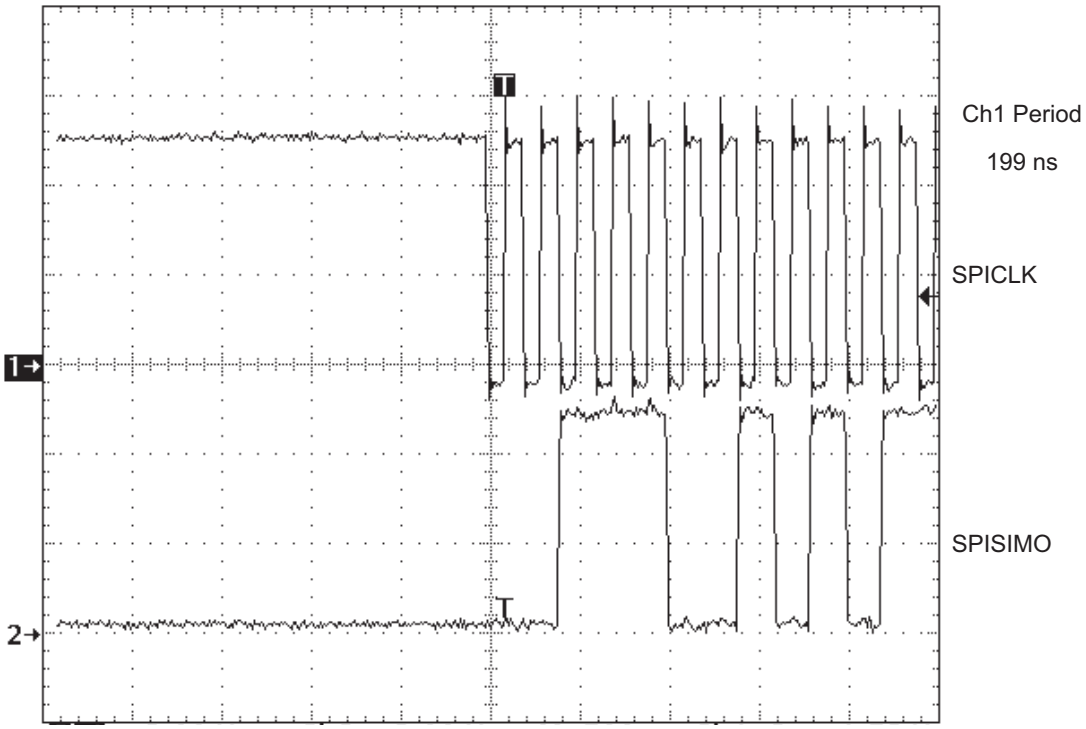
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-26. CLOCK POLARITY = 0, CLOCK PHASE = 1 (All data transitions are during the rising edge, but delayed by half clock cycle. Inactive level is low.)



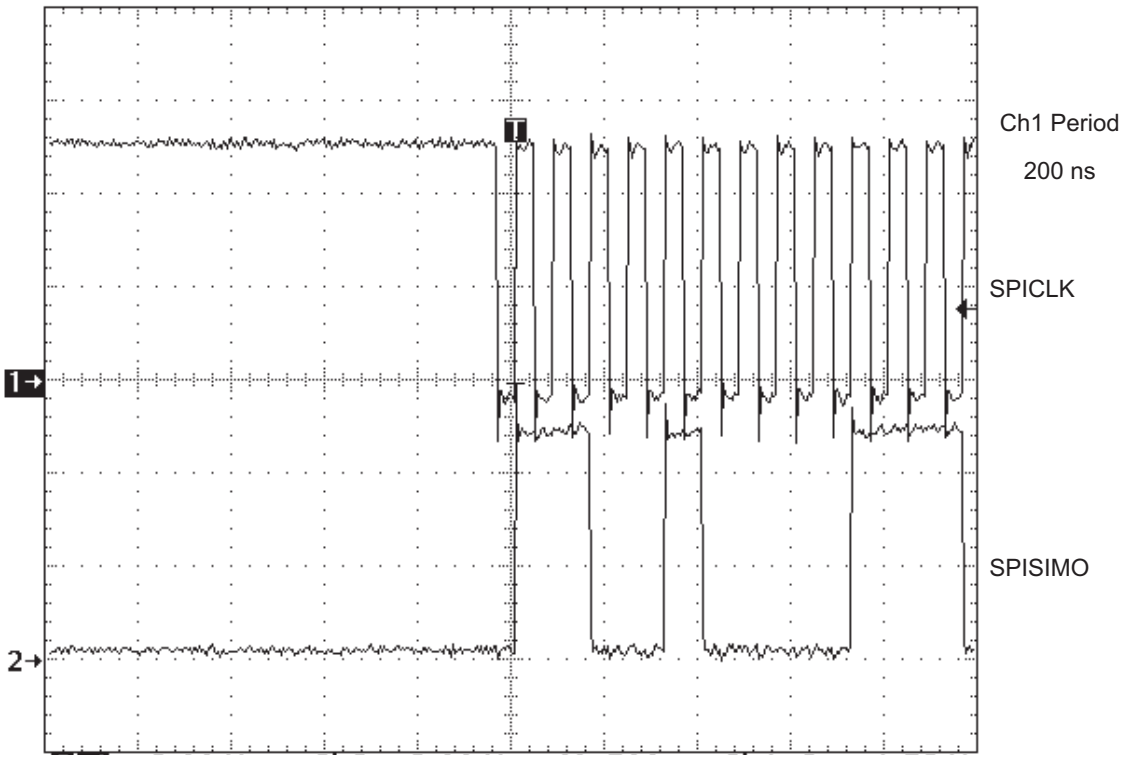
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-27. CLOCK POLARITY = 1, CLOCK PHASE = 0 (All data transitions are during the falling edge. Inactive level is high.)



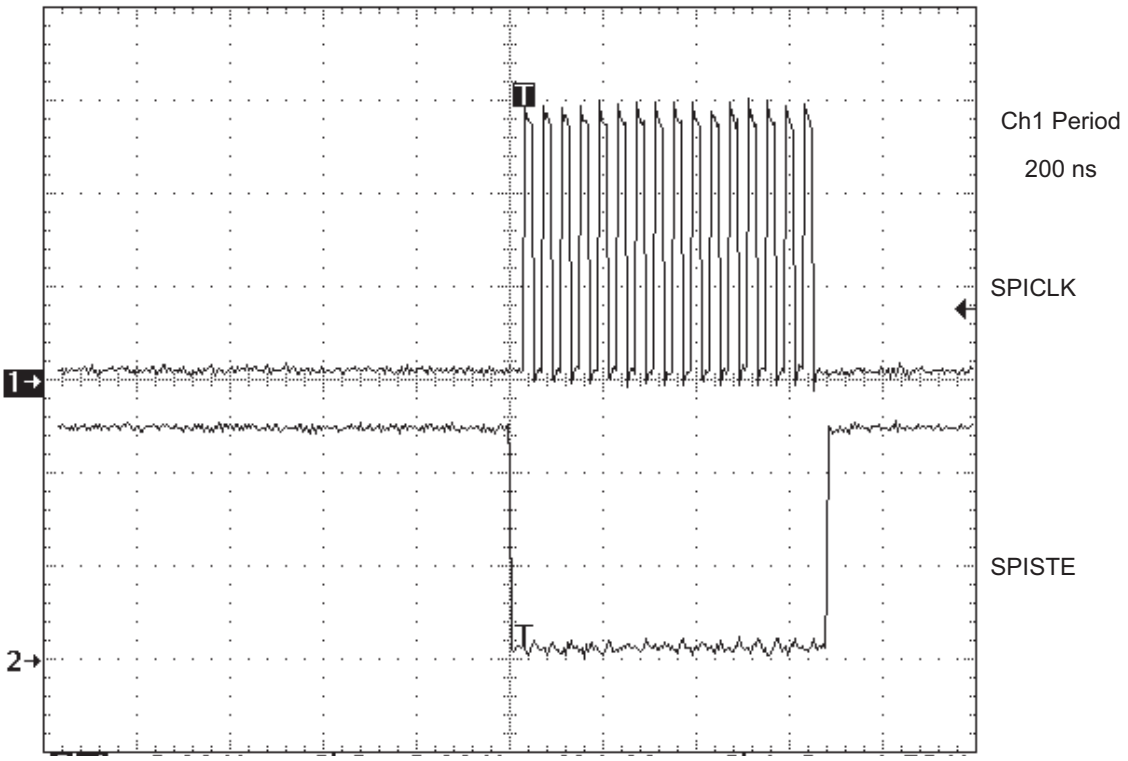
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-28. CLOCK POLARITY = 1, CLOCK PHASE = 1 (All data transitions are during the falling edge, but delayed by half clock cycle. Inactive level is high.)



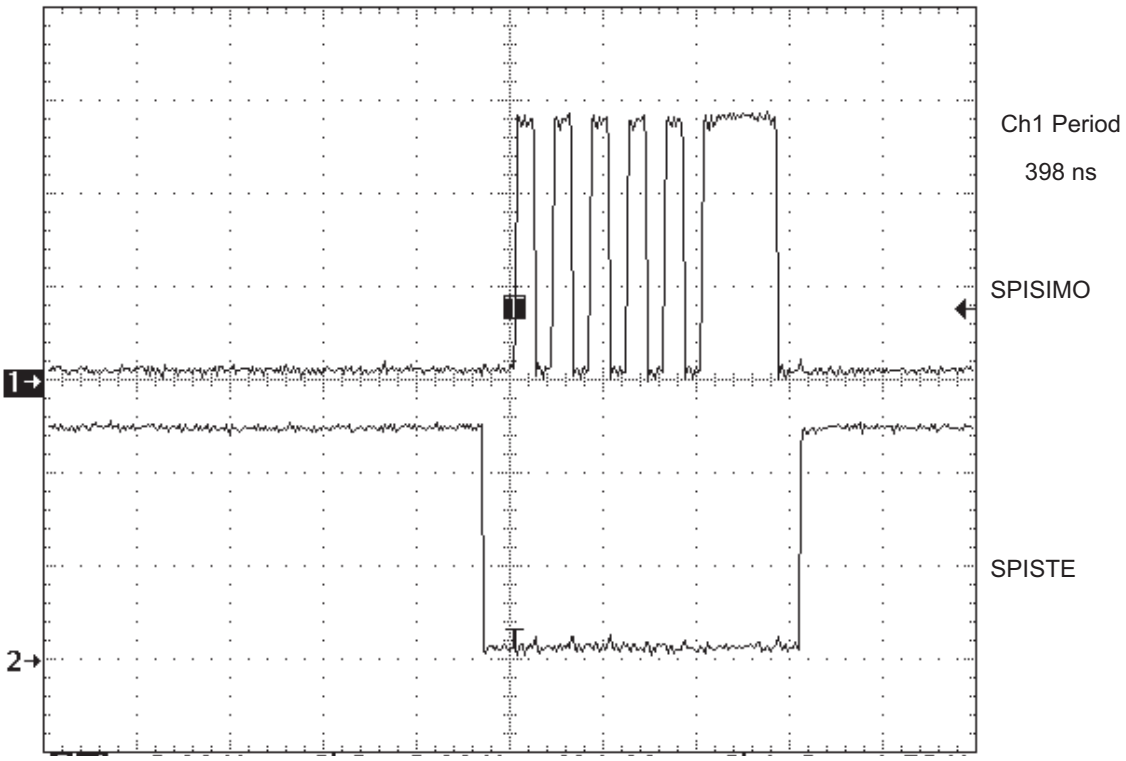
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-29. $\overline{\text{SPISTE}}$ Behavior in Master Mode (Master lowers $\overline{\text{SPISTE}}$ during the entire 16 bits of transmission.)



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 12-30. $\overline{\text{SPISTE}}$ Behavior in Slave Mode (Slave's $\overline{\text{SPISTE}}$ is lowered during the entire 16 bits of transmission.)



Copyright © 2012–2019, Texas Instruments Incorporated

C28 Serial Communications Interface (SCI)

This chapter describes the features and operation of the serial communication interface (SCI) module. SCI is a two−wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format. The SCI receiver and transmitter each have a 16 -level deep FIFO for reducing servicing overhead, and each has its own separate enable and interrupt bits. Both can be operated independently for half-duplex communication, or simultaneously for full-duplex communication.

To specify data integrity, the SCI checks received data for break detection, parity, overrun, and framing errors. The bit rate is programmable to different speeds through a 16-bit baud-select register.

Topic **Page**

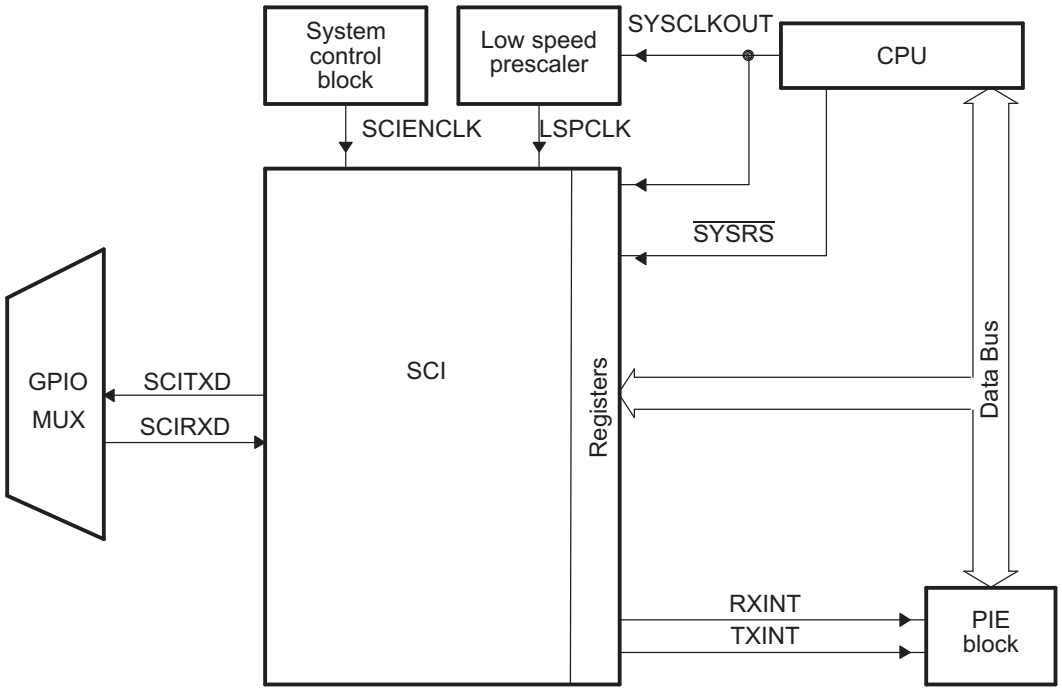
13.1 Enhanced SCI Module Overview **980**
13.2 C28 SCI-A to M3 UART4 Internal Loopback **992**
13.3 SCI Registers **994**

Copyright © 2012–2019, Texas Instruments Incorporated

13.1 Enhanced SCI Module Overview

The SCI interfaces are shown in Figure 13-1.

Figure 13-1. SCI CPU Interface



Features of the SCI module include:

- Two external pins:
  - SCITXD: SCI transmit-output pin
  - SCIRXD: SCI receive-input pin

  Both pins can be used as GPIO if not used for SCI.
- Baud rate programmable to 64K different rates
- Data-word format
  - One start bit
  - Data-word length programmable from one to eight bits
  - Optional even/odd/no parity bit
  - One or two stop bits
- Four error-detection flags: parity, overrun, framing, and break detection
- Two wake-up multiprocessor modes: idle-line and address bit
- Half- or full-duplex operation
- Double-buffered receive and transmit functions
- Transmitter and receiver operations can be accomplished through interrupt- driven or polled algorithms with status flags.
- Separate enable bits for transmitter and receiver interrupts (except BRKDT)
- NRZ (non-return-to-zero) format
- 13 SCI module control registers located in the control register frame beginning at address 7050h

  All registers in this module are 8-bit registers that are connected to Peripheral Frame 2. When a register is accessed, the register data is in the lower byte (7−0), and the upper byte (15−8) is read as zeros. Writing to the upper byte has no effect.

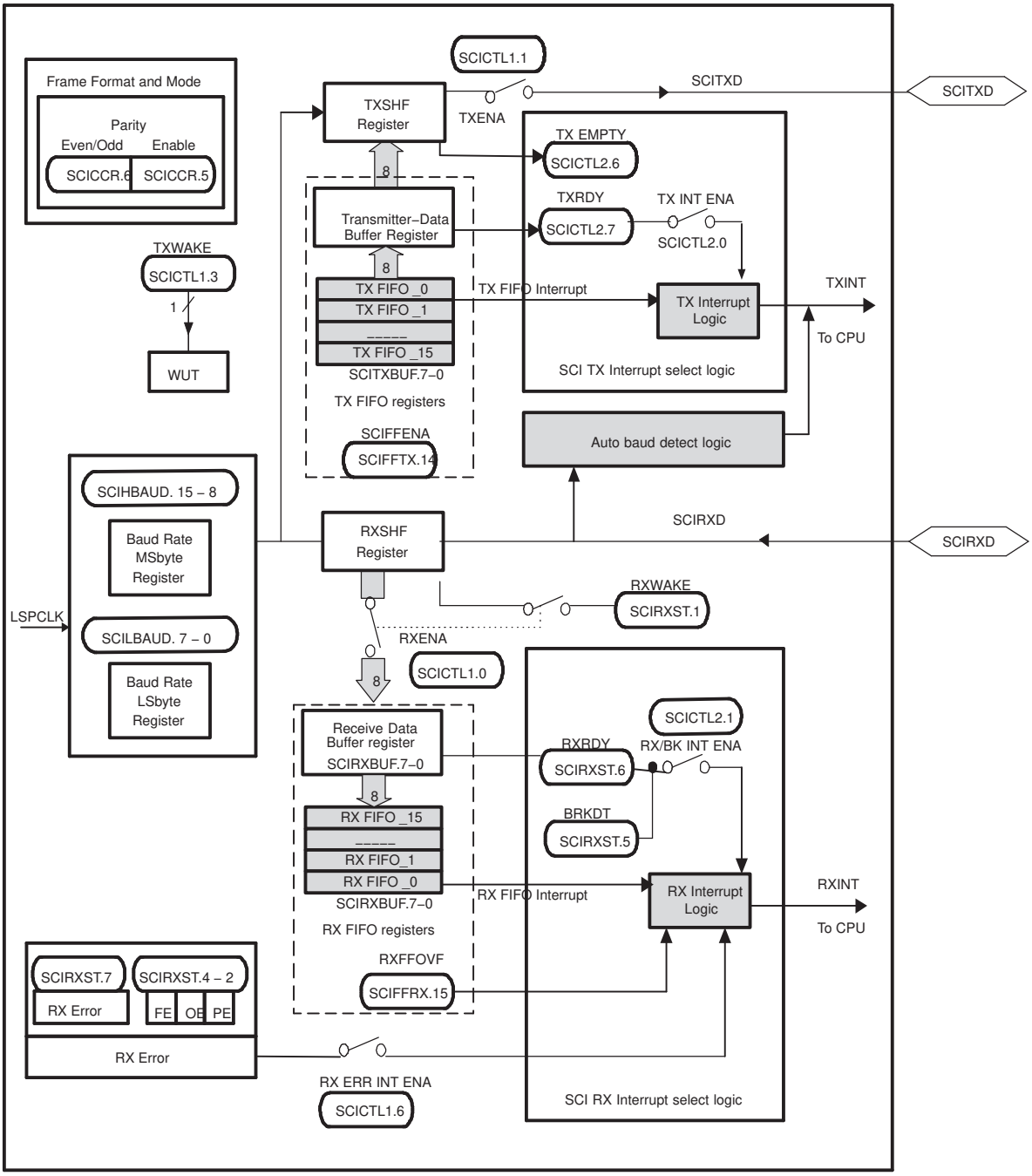
Enhanced features:

Copyright © 2012–2019, Texas Instruments Incorporated

- Auto-baud-detect hardware logic
- 16-level transmit/receive FIFO

Figure 13-2 shows the SCI module block diagram. The SCI port operation is configured and controlled by the registers listed in Table 13-1 and Table 13-2.

Figure 13-2. Serial Communications Interface (SCI) Module Block Diagram



Copyright © 2012–2019, Texas Instruments Incorporated

Table 13-1. SCI-A Registers

Name	Address Range	Size (x16)	Description
SCICCR	0x0000-7050	1	SCI-A Communications Control Register
SCICTL1	0x0000-7051	1	SCI-A Control Register 1
SCIHBAUD	0x0000-7052	1	SCI-A Baud Register, High Bits
SCILBAUD	0x0000-7053	1	SCI-A Baud Register, Low Bits
SCICTL2	0x0000-7054	1	SCI-A Control Register 2
SCIRXST	0x0000-7055	1	SCI-A Receive Status Register
SCIRXEMU	0x0000-7056	1	SCI-A Receive Emulation Data Buffer Register
SCIRXBUF	0x0000-7057	1	SCI-A Receive Data Buffer Register
SCITXBUF	0x0000-7059	1	SCI-A Transmit Data Buffer Register
SCIFFTX	0x0000-705A	1	SCI-A FIFO Transmit Register
SCIFFRX	0x0000-705B	1	SCI-A FIFO Receive Register
SCIFFCT	0x0000-705C	1	SCI-A FIFO Control Register
SCIPRI	0x0000-705F	1	SCI-A Priority Control Register

Table 13-2. SCI-B Registers

Name	Address Range	Size (x16)	Description[1] [2]
SCICCR	0x0000-7750	1	SCI-B Communications Control Register
SCICTL1	0x0000-7751	1	SCI-B Control Register 1
SCIHBAUD	0x0000-7752	1	SCI-B Baud Register, High Bits
SCILBAUD	0x0000-7753	1	SCI-B Baud Register, Low Bits
SCICTL2	0x0000-7754	1	SCI-B Control Register 2
SCIRXST	0x0000-7755	1	SCI-B Receive Status Register
SCIRXEMU	0x0000-7756	1	SCI-B Receive Emulation Data Buffer Register
SCIRXBUF	0x0000-7757	1	SCI-B Receive Data Buffer Register
SCITXBUF	0x0000-7759	1	SCI-B Transmit Data Buffer Register
SCIFFTX	0x0000-775A	1	SCI-B FIFO Transmit Register
SCIFFRX	0x0000-775B	1	SCI-B FIFO Receive Register
SCIFFCT	0x0000-775C	1	SCI-B FIFO Control Register
SCIPRI	0x0000-775F	1	SCI-B Priority Control Register

(1) The registers are mapped to peripheral frame 2. This frame allows only 16-bit accesses. Using 32-bit accesses will produce undefined results.
(2) SCIB is an optional peripheral. In some devices this may not be present. See the device-specific data sheet for peripheral availability.

13.1.1 Architecture

The major elements used in full-duplex operation are shown in Figure 13-2 and include:

- A transmitter (TX) and its major registers (upper half of Figure 13-2)
  - SCITXBUF — transmitter data buffer register. Contains data (loaded by the CPU) to be transmitted
  - TXSHF register — transmitter shift register. Accepts data from register SCITXBUF and shifts data onto the SCITXD pin, one bit at a time
- A receiver (RX) and its major registers (lower half of Figure 13-2)
  - RXSHF register — receiver shift register. Shifts data in from SCIRXD pin, one bit at a time
  - SCIRXBUF — receiver data buffer register. Contains data to be read by the CPU. Data from a remote processor is loaded into register RXSHF and then into registers SCIRXBUF and SCIRXEMU

Copyright © 2012–2019, Texas Instruments Incorporated

- A programmable baud generator
- Data-memory-mapped control and status registers

The SCI receiver and transmitter can operate either independently or simultaneously.

13.1.1.1 SCI Module Signal Summary

Table 13-3. SCI Module Signal Summary

Signal Name	Description
	External signals
SCIRXD	SCI Asynchronous Serial Port receive data
SCITXD	SCI Asynchronous Serial Port transmit data
	Control
Baud clock	LSPCLK Prescaled clock
	Interrupt signals
TXINT	Transmit interrupt
RXINT	Receive Interrupt

13.1.1.2 Multiprocessor and Asynchronous Communication Modes

The SCI has two multiprocessor protocols, the idle-line multiprocessor mode (see Section 13.1.1.5) and the address-bit multiprocessor mode (see Section 13.1.1.6). These protocols allow efficient data transfer between multiple processors.

The SCI offers the universal asynchronous receiver/transmitter (UART) communications mode for interfacing with many popular peripherals. The asynchronous mode (see Section 13.1.1.7) requires two lines to interface with many standard devices such as terminals and printers that use RS-232-C formats. Data transmission characteristics include:

- One start bit
- One to eight data bits
- An even/odd parity bit or no parity bit
- One or two stop bits

13.1.1.3 SCI Programmable Data Format

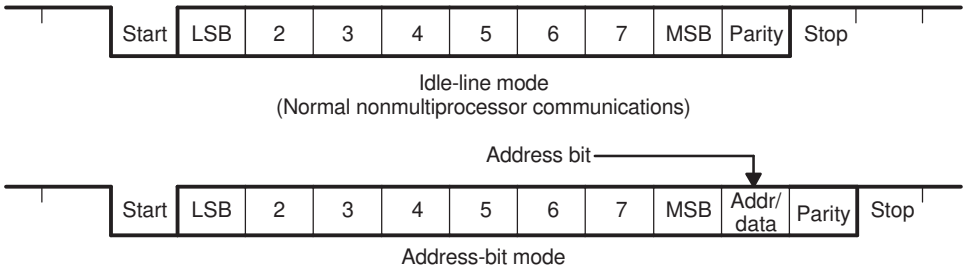
SCI data, both receive and transmit, is in NRZ (non-return-to-zero) format. The NRZ data format, shown in Figure 13-3, consists of:

- One start bit
- One to eight data bits
- An even/odd parity bit (optional)
- One or two stop bits
- An extra bit to distinguish addresses from data (address-bit mode only)

The basic unit of data is called a character and is one to eight bits in length. Each character of data is formatted with a start bit, one or two stop bits, and optional parity and address bits. A character of data with its formatting information is called a frame and is shown in Figure 13-3.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 13-3. Typical SCI Data Frame Formats



To program the data format, use the SCICCR register. The bits used to program the data format are shown in Table 13-4.

Table 13-4. Programming the Data Format Using SCICCR

Bit(s)	Bit Name	Designation	Functions
2-0	SCI CHAR2-0	SCICCR.2:0	Select the character (data) length (one to eight bits).
5	PARITY ENABLE	SCICCR.5	Enables the parity function if set to 1, or disables the parity function if cleared to 0.
6	EVEN/ODD PARITY	SCICCR.6	If parity is enabled, selects odd parity if cleared to 0 or even parity if set to 1.
7	STOP BITS	SCICCR.7	Determines the number of stop bits transmitted—one stop bit if cleared to 0 or two stop bits if set to 1.

13.1.1.4 SCI Multiprocessor Communication

The multiprocessor communication format allows one processor to efficiently send blocks of data to other processors on the same serial link. On one serial line, there should be only one transfer at a time. In other words, there can be only one talker on a serial line at a time.

Address Byte

The first byte of a block of information that the talker sends contains an address byte that is read by all listeners. Only listeners with the correct address can be interrupted by the data bytes that follow the address byte. The listeners with an incorrect address remain uninterrupted until the next address byte.

Sleep Bit

All processors on the serial link set the SCI SLEEP bit (bit 2 of SCICTL1) to 1 so that they are interrupted only when the address byte is detected. When a processor reads a block address that corresponds to the CPU device address as set by your application software, your program must clear the SLEEP bit to enable the SCI to generate an interrupt on receipt of each data byte.

Although the receiver still operates when the SLEEP bit is 1, it does not set RXRDY, RXINT, or any of the receiver error status bits to 1 unless the address byte is detected and the address bit in the received frame is a 1 (applicable to address-bit mode). The SCI does not alter the SLEEP bit; your software must alter the SLEEP bit.

13.1.1.4.1 Recognizing the Address Byte

A processor recognizes an address byte differently, depending on the multiprocessor mode used. For example:

- The idle-line mode (Section 13.1.1.5) leaves a quiet space before the address byte. This mode does not have an extra address/data bit and is more efficient than the address-bit mode for handling blocks that contain more than ten bytes of data. The idle-line mode should be used for typical non-multiprocessor SCI communication.
- The address-bit mode (Section 13.1.1.6) adds an extra bit (that is, an address bit) into every byte to distinguish addresses from data. This mode is more efficient in handling many small blocks of data

Copyright © 2012–2019, Texas Instruments Incorporated

because, unlike the idle mode, it does not have to wait between blocks of data. However, at a high transmit speed, the program is not fast enough to avoid a 10-bit idle in the transmission stream.

13.1.1.4.2 Controlling the SCI TX and RX Features

The multiprocessor mode is software selectable via the ADDR/IDLE MODE bit (SCICCR, bit 3). Both modes use the TXWAKE flag bit (SCICTL1, bit 3), RXWAKE flag bit (SCIRXST, bit1), and the SLEEP flag bit (SCICTL1, bit 2) to control the SCI transmitter and receiver features of these modes.

13.1.1.4.3 Receipt Sequence

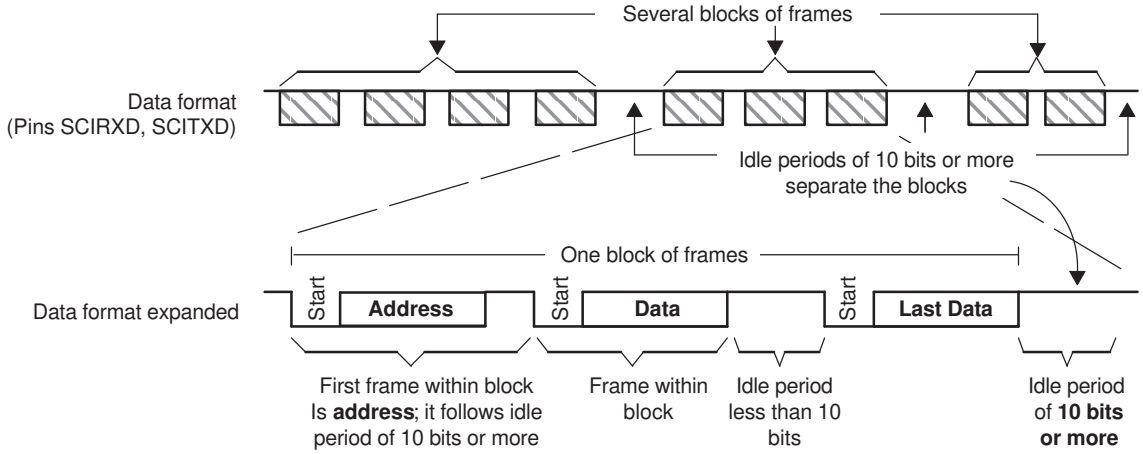
In both multiprocessor modes, the receive sequence is:

1. At the receipt of an address block, the SCI port wakes up and requests an interrupt (bit number 1 RX/BK INT ENA-of SCICTL2 must be enabled to request an interrupt). It reads the first frame of the block, which contains the destination address.
2. A software routine is entered through the interrupt and checks the incoming address. This address byte is checked against its device address byte stored in memory.
3. If the check shows that the block is addressed to the device CPU, the CPU clears the SLEEP bit and reads the rest of the block; if not, the software routine exits with the SLEEP bit still set and does not receive interrupts until the next block start.

13.1.1.5 Idle-Line Multiprocessor Mode

In the idle-line multiprocessor protocol (ADDR/IDLE MODE bit=0), blocks are separated by having a longer idle time between the blocks than between frames in the blocks. An idle time of ten or more high-level bits after a frame indicates the start of a new block. The time of a single bit is calculated directly from the baud value (bits per second). The idle-line multiprocessor communication format is shown in Figure 13-4 (ADDR/IDLE MODE bit is bit 3 of SCICCR).

Figure 13-4. Idle-Line Multiprocessor Communication Format



13.1.1.5.1 Idle-Line Mode Steps

The steps followed by the idle-line mode:

Step 1. SCI wakes up after receipt of the block-start signal.

Step 2. The processor recognizes the next SCI interrupt.

Step 3. The interrupt service routine compares the received address (sent by a remote transmitter) to its own.

Step 4. If the CPU is being addressed, the service routine clears the SLEEP bit and receives the rest of the data block.

Step 5. If the CPU is not being addressed, the SLEEP bit remains set. This lets the CPU continue to execute its main program without being interrupted by the SCI port until the next detection of

Copyright © 2012–2019, Texas Instruments Incorporated

a block start.

13.1.1.5.2 Block Start Signal

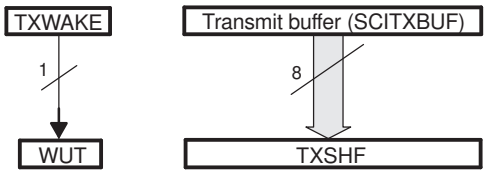
There are two ways to send a block-start signal:

1. **Method 1:** Deliberately leave an idle time of ten bits or more by delaying the time between the transmission of the last frame of data in the previous block and the transmission of the address frame of the new block.
2. **Method 2:** The SCI port first sets the TXWAKE bit (SCICTL1, bit 3) to 1 before writing to the SCITXBUF register. This sends an idle time of exactly 11 bits. In this method, the serial communications line is not idle any longer than necessary. (A don't care byte has to be written to SCITXBUF after setting TXWAKE, and before sending the address, so as to transmit the idle time.)

13.1.1.5.3 Wake-UP Temporary (WUT) Flag

Associated with the TXWAKE bit is the wake-up temporary (WUT) flag. WUT is an internal flag, double-buffered with TXWAKE. When TXSHF is loaded from SCITXBUF, WUT is loaded from TXWAKE, and the TXWAKE bit is cleared to 0. This arrangement is shown in Figure 13-5.

Figure 13-5. Double-Buffered WUT and TXSHF



A WUT = wake-up temporary

Sending a Block Start Signal

To send out a block-start signal of exactly one frame time during a sequence of block transmissions:

1. Write a 1 to the TXWAKE bit.
2. Write a data word (content not important: a don't care) to the SCITXBUF register (transmit data buffer) to send a block-start signal. (The first data word written is suppressed while the block-start signal is sent out and ignored after that.) When the TXSHF (transmit shift register) is free again, SCITXBUF contents are shifted to TXSHF, the TXWAKE value is shifted to WUT, and then TXWAKE is cleared.

   Because TXWAKE was set to a 1, the start, data, and parity bits are replaced by an idle period of 11 bits transmitted following the last stop bit of the previous frame.
3. **Write a new address value to SCITXBUF**

   A don't-care data word must first be written to register SCITXBUF so that the TXWAKE bit value can be shifted to WUT. After the don't-care data word is shifted to the TXSHF register, the SCITXBUF (and TXWAKE if necessary) can be written to again because TXSHF and WUT are both double-buffered.

13.1.1.5.4 Receiver Operation

The receiver operates regardless of the SLEEP bit. However, the receiver neither sets RXRDY nor the error status bits, nor does it request a receive interrupt until an address frame is detected.

13.1.1.6 Address-Bit Multiprocessor Mode

In the address-bit protocol (ADDR/IDLE MODE bit=1), frames have an extra bit called an address bit that immediately follows the last data bit. The address bit is set to 1 in the first frame of the block and to 0 in all other frames. The idle period timing is irrelevant (see Figure 13-6).

Copyright © 2012–2019, Texas Instruments Incorporated

13.1.1.6.1 Sending an Address

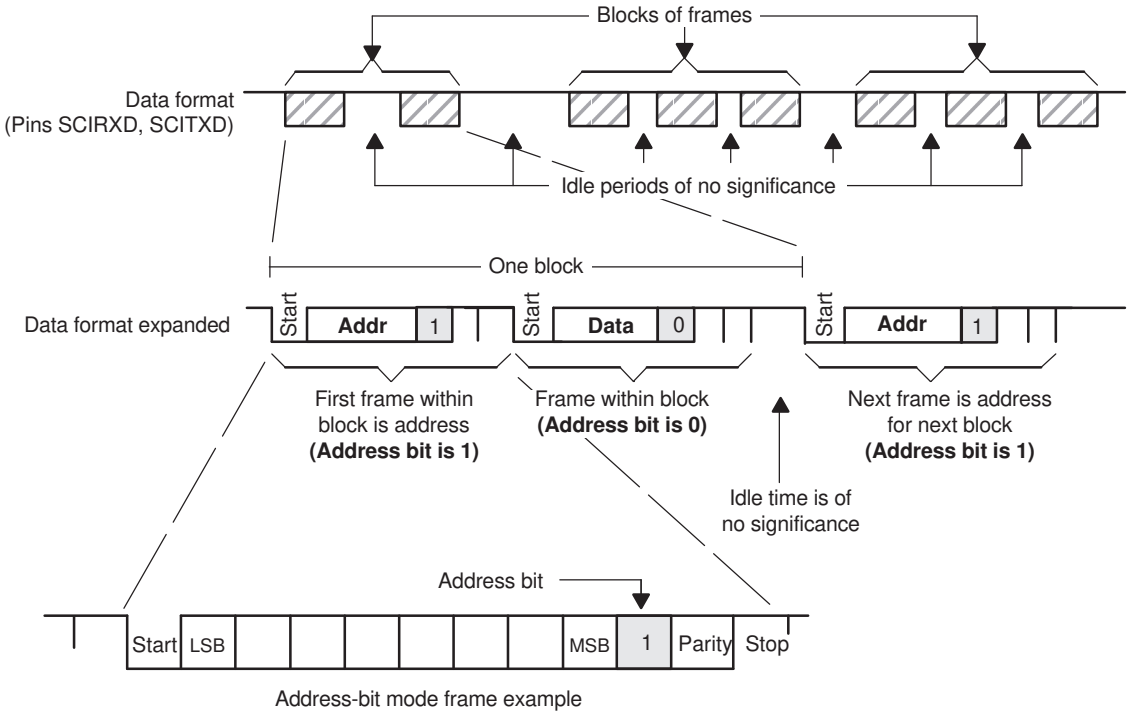
The TXWAKE bit value is placed in the address bit. During transmission, when the SCITXBUF register and TXWAKE are loaded into the TXSHF register and WUT respectively, TXWAKE is reset to 0 and WUT becomes the value of the address bit of the current frame. Thus, to send an address:

1. Set the TXWAKE bit to 1 and write the appropriate address value to the SCITXBUF register.

   When this address value is transferred to the TXSHF register and shifted out, its address bit is sent as a 1. This flags the other processors on the serial link to read the address.
2. Write to SCITXBUF and TXWAKE after TXSHF and WUT are loaded. (Can be written to immediately since both TXSHF and WUT are both double-buffered.
3. Leave the TXWAKE bit set to 0 to transmit non-address frames in the block.

NOTE: As a general rule, the address-bit format is typically used for data frames of 11 bytes or less. This format adds one bit value (1 for an address frame, 0 for a data frame) to all data bytes transmitted. The idle-line format is typically used for data frames of 12 bytes or more.

Figure 13-6. Address-Bit Multiprocessor Communication Format



13.1.1.7 SCI Communication Format

The SCI asynchronous communication format uses either single line (one way) or two line (two way) communications. In this mode, the frame consists of a start bit, one to eight data bits, an optional even/odd parity bit, and one or two stop bits (shown in Figure 13-7). There are eight SCICLK periods per data bit.

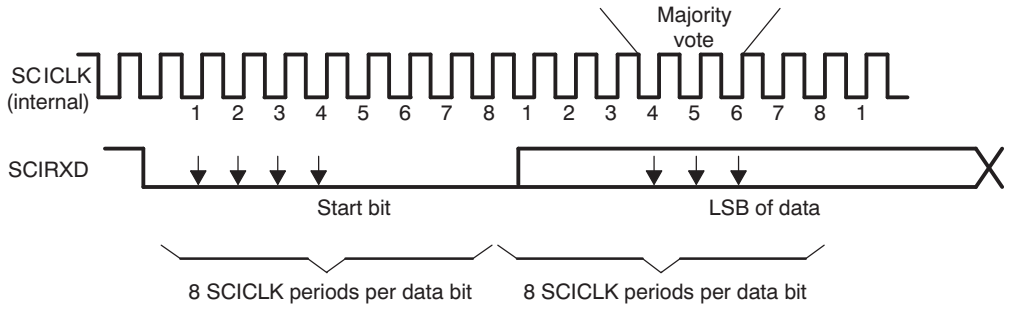
The receiver begins operation on receipt of a valid start bit. A valid start bit is identified by four consecutive internal SCICLK periods of zero bits as shown in Figure 13-7. If any bit is not zero, then the processor starts over and begins looking for another start bit.

For the bits following the start bit, the processor determines the bit value by making three samples in the middle of the bits. These samples occur on the fourth, fifth, and sixth SCICLK periods, and bit-value determination is on a majority (two out of three) basis. Figure 13-7 illustrates the asynchronous communication format for this with a start bit showing where a majority vote is taken.

Copyright © 2012–2019, Texas Instruments Incorporated

Since the receiver synchronizes itself to frames, the external transmitting and receiving devices do not have to use a synchronized serial clock. The clock can be generated locally.

Figure 13-7. SCI Asynchronous Communications Format

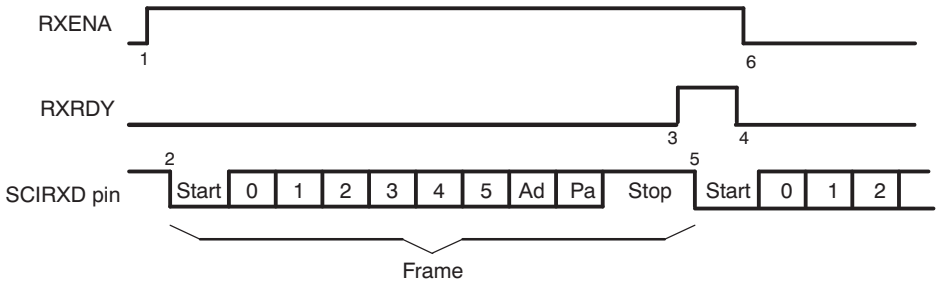


13.1.1.7.1 Receiver Signals in Communication Modes

Figure 13-8 illustrates an example of receiver signal timing that assumes the following conditions:

- Address-bit wake-up mode (address bit does not appear in idle-line mode)
- Six bits per character

Figure 13-8. SCI RX Signals in Communication Modes



(1) Data arrives on the SCIRXD pin, start bit detected.

(2) Bit RXENA is brought low to disable the receiver. Data continues to be assembled in RXSHF but is not transferred to the receiver buffer register.

Notes:

1. Flag bit RXENA (SCICTL1, bit 0) goes high to enable the receiver.
2. Data arrives on the SCIRXD pin, start bit detected.
3. Data is shifted from RXSHF to the receiver buffer register (SCIRXBUF); an interrupt is requested. Flag bit RXRDY (SCIRXST, bit 6) goes high to signal that a new character has been received.
4. The program reads SCIRXBUF; flag RXRDY is automatically cleared.
5. The next byte of data arrives on the SCIRXD pin; the start bit is detected, then cleared.
6. Bit RXENA is brought low to disable the receiver. Data continues to be assembled in RXSHF but is not transferred to the receiver buffer register.

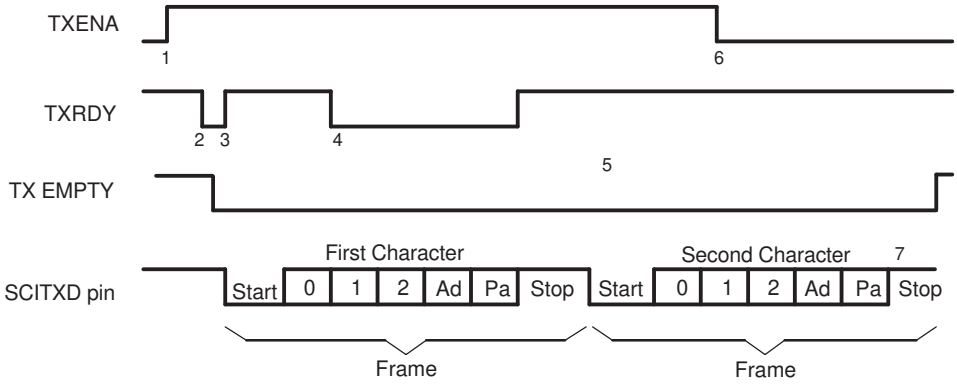
13.1.1.7.2 Transmitter Signals in Communication Modes

Figure 13-9 illustrates an example of transmitter signal timing that assumes the following conditions:

- Address-bit wake-up mode (address bit does not appear in idle-line mode)
- Three bits per character

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 13-9. SCI TX Signals in Communications Mode



Notes:

1. Bit TXENA (SCICTL1, bit 1) goes high, enabling the transmitter to send data.
2. SCITXBUF is written to; thus, (1) the transmitter is no longer empty, and (2) TXRDY goes low.
3. The SCI transfers data to the shift register (TXSHF). The transmitter is ready for a second character (TXRDY goes high), and it requests an interrupt (to enable an interrupt, bit TX INT ENA — SCICTL2, bit 0 — must be set).
4. The program writes a second character to SCITXBUF after TXRDY goes high (item 3). (TXRDY goes low again after the second character is written to SCITXBUF.)
5. Transmission of the first character is complete. Transfer of the second character to shift register TXSHF begins.
6. Bit TXENA goes low to disable the transmitter; the SCI finishes transmitting the current character.
7. Transmission of the second character is complete; transmitter is empty and ready for new character.

13.1.1.8 SCI Port Interrupts

The SCI receiver and transmitter can be interrupt controlled. The SCICTL2 register has one flag bit (TXRDY) that indicates active interrupt conditions, and the SCIRXST register has two interrupt flag bits (RXRDY and BRKDT), plus the RX ERROR interrupt flag which is a logical OR of the FE, OE, BRKDT, and PE conditions. The transmitter and receiver have separate interrupt-enable bits. When not enabled, the interrupts are not asserted; however, the condition flags remain active, reflecting transmission and receipt status.

The SCI has independent peripheral interrupt vectors for the receiver and transmitter. Peripheral interrupt requests can be either high priority or low priority. This is indicated by the priority bits which are output from the peripheral to the PIE controller. When both RX and TX interrupt requests are made at the same priority level, the receiver always has higher priority than the transmitter, reducing the possibility of receiver overrun.

The operation of peripheral interrupts is described in the peripheral interrupt expansion controller chapter of the *TMS320x2833x, 2823x System Control and Interrupts Peripheral Reference Guide* (literature number SPRUFB0 .

- If the RX/BK INT ENA bit (SCICTL2, bit 1) is set, the receiver peripheral interrupt request is asserted when one of the following events occurs:
  - The SCI receives a complete frame and transfers the data in the RXSHF register to the SCIRXBUF register. This action sets the RXRDY flag (SCIRXST, bit 6) and initiates an interrupt.
  - A break detect condition occurs (the SCIRXD is low for ten bit periods following a missing stop bit). This action sets the BRKDT flag bit (SCIRXST, bit 5) and initiates an interrupt.
- If the TX INT ENA bit (SCICTL2.0) is set, the transmitter peripheral interrupt request is asserted whenever the data in the SCITXBUF register is transferred to the TXSHF register, indicating that the CPU can write to SCITXBUF; this action sets the TXRDY flag bit (SCICTL2, bit 7) and initiates an interrupt.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: Interrupt generation due to the RXRDY and BRKDT bits is controlled by the RX/BK INT ENA bit (SCICTL2, bit 1). Interrupt generation due to the RX ERROR bit is controlled by the RX ERR INT ENA bit (SCICTL1, bit 6).

13.1.1.9 SCI Baud Rate Calculations

The internally generated serial clock is determined by the low-speed peripheral clock LSPCLK) and the baud-select registers. The SCI uses the 16-bit value of the baud-select registers to select one of the 64K different serial clock rates possible for a given LSPCLK.

See the bit descriptions in Section 13.3.4, for the formula to use when calculating the SCI asynchronous baud. Table 13-5 shows the baud-select values for common SCI bit rates.

Table 13-5. Asynchronous Baud Register Values for Common SCI Bit Rates

		LSPCLK Clock Frequency, 37.5 MHz	
Ideal Baud	**BRR**	**Actual Baud**	**% Error**
2400	1952 (7A0h)	2400	0
4800	976 (3D0h)	4798	-0.04
9600	487 (1E7h)	9606	0.06
19200	243 (F3h)	19211	0.06
38400	121 (79h)	38422	0.06

13.1.1.10 SCI Enhanced Features

The 28x SCI features autobaud detection and transmit/receive FIFO. The following section explains the FIFO operation.

13.1.1.10.1 SCI FIFO Description

The following steps explain the FIFO features and help with programming the SCI with FIFOs.

1. *Reset.* At reset the SCI powers up in standard SCI mode and the FIFO function is disabled. The FIFO registers SCIFFTX, SCIFFRX, and SCIFFCT remain inactive.
2. *Standard SCI.* The standard F24x SCI modes will work normally with TXINT/RXINT interrupts as the interrupt source for the module.
3. *FIFO enable.* FIFO mode is enabled by setting the SCIFFEN bit in the SCIFFTX register. SCIRST can reset the FIFO mode at any stage of its operation.
4. Active registers. All the SCI registers and SCI FIFO registers (SCIFFTX, SCIFFRX, and SCIFFCT) are active.
5. *Interrupts.* FIFO mode has two interrupts; one for transmit FIFO, TXINT and one for receive FIFO, RXINT. RXINT is the common interrupt for SCI FIFO receive, receive error, and receive FIFO overflow conditions. The TXINT of the standard SCI will be disabled and this interrupt will service as SCI transmit FIFO interrupt.
6. *Buffers.* Transmit and receive buffers are supplemented with two 16-level FIFOs. The transmit FIFO registers are 8 bits wide and receive FIFO registers are 10 bits wide. The one-word transmit buffer of the standard SCI, functions as a transition buffer between the transmit FIFO and shift register. The one word transmit buffer is loaded from transmit FIFO only after the last bit of the shift register is shifted out. With the FIFO enabled, TXSHF is directly loaded after an optional delay value (SCIFFCT), TXBUF is not used. When FIFO mode is enabled for SCI, characters written to SCITXBUF are queued in to SCI-TXFIFO and the characters received in SCI-RXFIFO can be read using SCIRXBUF.
7. *Delayed transfer.* The rate at which words in the FIFO are transferred to the transmit shift register is programmable. The SCIFFCT register bits (7−0) FFTXDLY7−FFTXDLY0 define the delay between the word transfer. The delay is defined in the number SCI baud clock cycles. The 8 bit register can define a minimum delay of 0 baud clock cycles and a maximum of 256-baud clock cycles. With zero delay, the SCI module can transmit data in continuous mode with the FIFO words shifting out back to back.

Copyright © 2012–2019, Texas Instruments Incorporated

With the 256 clock delay the SCI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 256 baud clocks between each words. The programmable delay facilitates communication with slow SCI/UARTs with little CPU intervention.

8. *FIFO status bits.* Both the transmit and receive FIFOs have status bits TXFFST or RXFFST (bits 12−0) that define the number of words available in the FIFOs at any time. The transmit FIFO reset bit TXFIFO and receive reset bit RXFIFO reset the FIFO pointers to zero when these bits are cleared to 0. The FIFOs resumes operation from start once these bits are set to one.
9. *Programmable interrupt levels.* Both transmit and receive FIFO can generate CPU interrupts. The interrupt trigger is generated whenever the transmit FIFO status bits TXFFST (bits 12−8) match (less than or equal to) the interrupt trigger level bits TXFFIL (bits 4−0). This provides a programmable interrupt trigger for transmit and receive sections of the SCI. Default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO, respectively.

Figure 13-10 and Table 13-6 explain the operation/configuration of SCI interrupts in nonFIFO/FFO mode.

Figure 13-10. SCI FIFO Interrupt Flags and Enable Logic

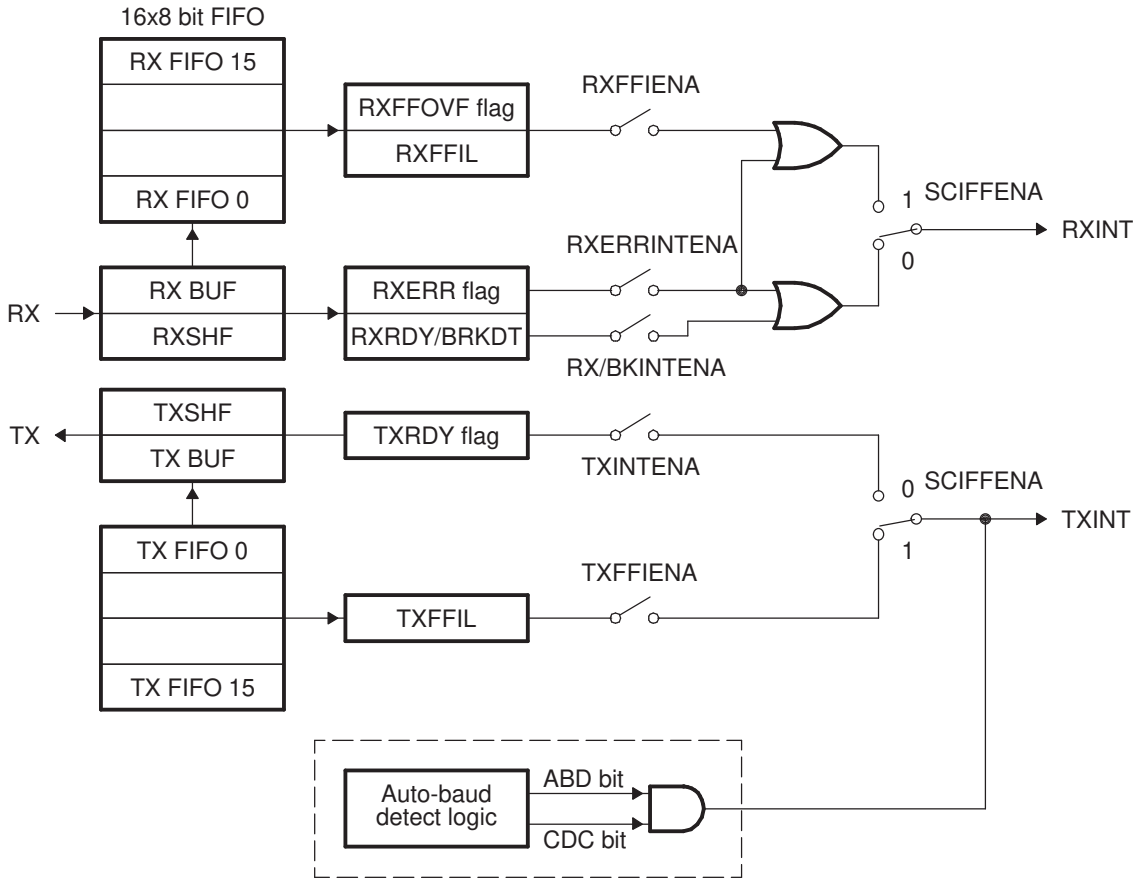


Table 13-6. SCI Interrupt Flags

FIFO Options[1]	SCI Interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable SCIFFENA	Interrupt Line
SCI without FIFO	Receive error	RXERR [2]	RXERRINTENA	0	RXINT
	Receive break	BRKDT	RX/BKINTENA	0	RXINT
	Data receive	RXRDY	RX/BKINTENA	0	RXINT
	Transmit empty	TXRDY	TXINTENA	0	TXINT
SCI with FIFO	Receive error and receive break	RXERR	RXERRINTENA	1	RXINT

(1) FIFO mode TXSHF is directly loaded after delay value, TXBUF is not used.
(2) RXERR can be set by BRKDT, FE, OE, PE flags. In FIFO mode, BRKDT interrupt is only through RXERR flag

Copyright © 2012–2019, Texas Instruments Incorporated

Table 13-6. SCI Interrupt Flags (continued)

FIFO Options[1]	SCI Interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable SCIFFENA	Interrupt Line
	FIFO receive	RXFFIL	RXFFIENA	1	RXINT
	Transmit empty	TXFFIL	TXFFIENA	1	TXINT
Auto-baud	Auto-baud detected	ABD	Don't care	x	TXINT

13.1.1.10.2 SCI Auto-Baud

Most SCI modules do not have an auto-baud detect logic built-in hardware. These SCI modules are integrated with embedded controllers whose clock rates are dependent on PLL reset values. Often embedded controller clocks change after final design. In the enhanced feature set this module supports an autobaud-detect logic in hardware. The following section explains the enabling sequence for autobaud-detect feature.

13.1.1.10.3 Autobaud-Detect Sequence

Bits ABD and CDC in SCIFFCT control the autobaud logic. The SCIRST bit should be enabled to make autobaud logic work.

If ABD is set while CDC is 1, which indicates auto-baud alignment, SCI transmit FIFO interrupt will occur (TXINT). After the interrupt service CDC bit has to be cleared by software. If CDC remains set even after interrupt service, there should be no repeat interrupts.

1. Enable autobaud-detect mode for the SCI by setting the CDC bit (bit 13) in SCIFFCT and clearing the ABD bit (Bit 15) by writing a 1 to ABDCLR bit (bit 14).
2. Initialize the baud register to be 1 or less than a baud rate limit of 500 Kbps.
3. Allow SCI to receive either character "A" or "a" from a host at the desired baud rate. If the first character is either "A" or "a". the autobaud- detect hardware will detect the incoming baud rate and set the ABD bit.
4. The auto-detect hardware will update the baud rate register with the equivalent baud value hex. The logic will also generate an interrupt to the CPU.
5. Respond to the interrupt clear ADB bit by writing a 1 to ABD CLR (bit 14) of SCIFFCT register and disable further autobaud locking by clearing CDC bit by writing a 0.
6. Read the receive buffer for character "A" or "a" to empty the buffer and buffer status.
7. If ABD is set while CDC is 1, which indicates autobaud alignment, the SCI transmit FIFO interrupt will occur (TXINT). After the interrupt service CDC bit must be cleared by software.

NOTE: At higher baud rates, the slew rate of the incoming data bits can be affected by transceiver and connector performance. While normal serial communications may work well, this slew rate may limit reliable autobaud detection at higher baud rates (typically beyond 100k baud) and cause the auto-baudlock feature to fail.

To avoid this, the followng is recommended:

- Achieve a baud-lock between the host and 28x SCI boot loader using a lower baud rate.
- The host may then handshake with the loaded 28x application to set the SCI baud rate register to the desired higher baud rate.

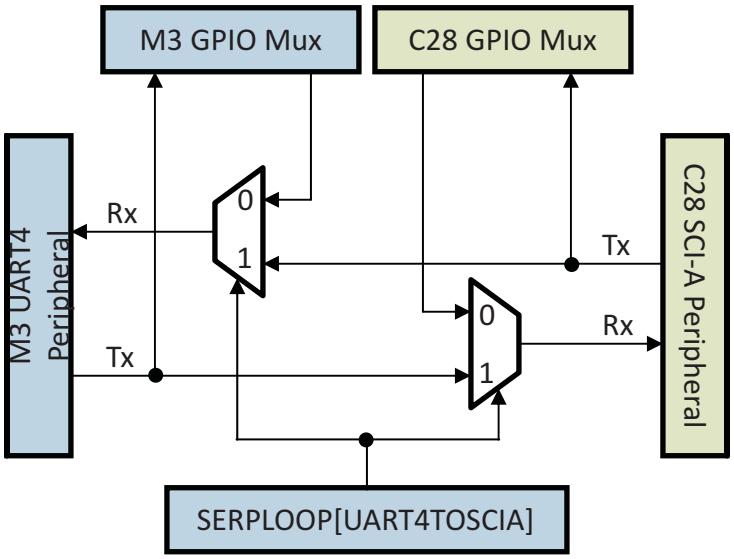
13.2 C28 SCI-A to M3 UART4 Internal Loopback

The C28 SCI-A peripheral can be internally connected to the M3 UART4 peripheral. External GPIO pins are not used when the loopback feature is enabled and can be used for other functions.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 13-11 illustrates the loopback connections between the C28 SCI-A and M3 UART4. The C28 SCI-ATx is connected to the M3 UART4Rx and C28 SCI-ARx is connected to M3 UART4Tx when loopback mode is enabled. Loopback is enabled by setting the UART4TOSCIA bit in the M3 SERPLOOP register. For a complete description of the SERPLOOP register, refer to the *System Control and Interrupt* chapter.

Figure 13-11. UART and SCI Connections for Loopback Mode



13.2.1 Loopback Initialization and Configuration

To enable M3 UART4 to C28 SCI-A loopback, follow these steps:

1. Enable and configure the M3 UART4 module by following the steps outlined in the Initialization and Configuration section of the *M3 Universal Asynchronous Receivers/Transmitters (UARTs)* chapter.
2. Enable internal loopback mode by setting SERPLOOP[UART4TOSCIA] = 1 on the M3 subsystem.
3. Enable and configure the C28 SCI-A module as described in Section 13.1.

To disable loopback between the M3 and C28, set SERPLOOP[UART4TOSCIA] = 0.

Copyright © 2012–2019, Texas Instruments Incorporated

13.3 SCI Registers

The functions of the SCI are software configurable. Sets of control bits, organized into dedicated bytes, are programmed to initialize the desired SCI communications format. This includes operating mode and protocol, baud value, character length, even/odd parity or no parity, number of stop bits, and interrupt priorities and enables.

13.3.1 SCI Module Register Summary

The SCI is controlled and accessed through registers listed in Table 13-7 and Table 13-8, which are described in the sections that follow.

Table 13-7. SCIA Registers

Register Mnemonic	Address	Number of Bits	Description
SCICCR	0x0000-7050	1	SCI-A Communications Control Register
SCICTL1	0x0000-7051	1	SCI-A Control Register 1
SCIHBAUD	0x0000-7052	1	SCI-A Baud Register, High Bits
SCILBAUD	0x0000-7053	1	SCI-A Baud Register, Low Bits
SCICTL2	0x0000-7054	1	SCI-A Control Register 2
SCIRXST	0x0000-7055	1	SCI-A Receive Status Register
SCIRXEMU	0x0000-7056	1	SCI-A Receive Emulation Data Buffer Register
SCIRXBUF	0x0000-7057	1	SCI-A Receive Data Buffer Register
SCITXBUF	0x0000-7059	1	SCI-A Transmit Data Buffer Register
SCIFFTX (1)	0x0000-705A	1	SCI-A FIFO Transmit Register
SCIFFRX (1)	0x0000-705B	1	SCI-A FIFO Receive Register
SCIFFCT (1)	0x0000-705C	1	SCI-A FIFO Control Register
SCIPRI	0x0000-705F	1	SCI-A Priority Control Register

(1) These registers operate in enhanced mode.

Table 13-8. SCIB Registers

Name	Address Range	Number of Bits	Description
SCICCR	0x0000-7750	1	SCI-B Communications Control Register
SCICTL1	0x0000-7751	1	SCI-B Control Register 1
SCIHBAUD	0x0000-7752	1	SCI-B Baud Register, High Bits
SCILBAUD	0x0000-7753	1	SCI-B Baud Register, Low Bits
SCICTL2	0x0000-7754	1	SCI-B Control Register 2
SCIRXST	0x0000-7755	1	SCI-B Receive Status Register
SCIRXEMU	0x0000-7756	1	SCI-B Receive Emulation Data Buffer Register
SCIRXBUF	0x0000-7757	1	SCI-B Receive Data Buffer Register
SCITXBUF	0x0000-7759	1	SCI-B Transmit Data Buffer Register
SCIFFTX	0x0000-775A	1	SCI-B FIFO Transmit Register
SCIFFRX	0x0000-775B	1	SCI-B FIFO Receive Register
SCIFFCT	0x0000-775C	1	SCI-B FIFO Control Register
SCIPRI	0x0000-775F	1	SCI-B Priority Control Register

Copyright © 2012–2019, Texas Instruments Incorporated

13.3.2 SCI Communication Control Register (SCICCR)

SCICCR defines the character format, protocol, and communications mode used by the SCI.

Figure 13-12. SCI Communication Control Register (SCICCR) — Address 7050h

7	6	5	4	3	2	1	0
STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	LOOPBACK ENA	ADDR/IDLE MODE	SCICHAR2	SCICHAR1	SCICHAR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-9. SCI Communication Control Register (SCICCR) Field Descriptions

Bit	Field	Value	Description
7	STOP BITS		SCI number of stop bits. This bit specifies the number of stop bits transmitted. The receiver checks for only one stop bit.
		0	One stop bit
		1	Two stop bits
6	EVEN/ODD PARITY		SCI parity odd/even selection. If the PARITY ENABLE bit (SCICCR, bit 5) is set, PARITY (bit 6) designates odd or even parity (odd or even number of bits with the value of 1 in both transmitted and received characters).
		0	Odd parity
		1	Even parity
5	PARITY ENABLE		SCI parity enable. This bit enables or disables the parity function. If the SCI is in the address-bit multiprocessor mode (set using bit 3 of this register), the address bit is included in the parity calculation (if parity is enabled). For characters of less than eight bits, the remaining unused bits should be masked out of the parity calculation.
		0	Parity disabled; no parity bit is generated during transmission or is expected during reception
		1	Parity is enabled
4	LOOP BACK ENA		Loop Back test mode enable. This bit enables the Loop Back test mode where the Tx pin is internally connected to the Rx pin.
		0	Loop Back test mode disabled
		1	Loop Back test mode enabled
3	ADDR/IDLE MODE		SCI multiprocessor mode control bit. This bit selects one of the multiprocessor protocols. Multiprocessor communication is different from the other communication modes because it uses SLEEP and TXWAKE functions (bits SCICTL1, bit 2 and SCICTL1, bit 3, respectively). The idle-line mode is usually used for normal communications because the address-bit mode adds an extra bit to the frame. The idle-line mode does not add this extra bit and is compatible with RS-232 type communications.
		0	Idle-line mode protocol selected
		1	Address-bit mode protocol selected
2-0	SCI CHAR2−0		Character-length control bits 2−0. These bits select the SCI character length from one to eight bits. Characters of less than eight bits are right-justified in SCIRXBUF and SCIRXEMU and are padded with leading zeros in SCIRXBUF. SCITXBUF doesn't need to be padded with leading zeros. The bit values and character lengths for SCI CHAR2-0 bits are as follows:

SCI CHAR2−0 Bit Values (Binary)

SCI CHAR2	SCI CHAR1	SCI CHAR0	Character Length (Bits)
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

Copyright © 2012–2019, Texas Instruments Incorporated

13.3.3 SCI Control Register 1 (SCICTL1)

SCICTL1 controls the receiver/transmitter enable, TXWAKE and SLEEP functions, and the SCI software reset.

Figure 13-13. SCI Control Register 1 (SCICTL1) — Address 7051h

7	6	5	4	3	2	1	0
Reserved	RX ERR INT ENA	SW RESET	Reserved	TXWAKE	SLEEP	TXENA	RXENA
R-0	R/W-0	R/W-0	R-0	R/S-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-10. SCI Control Register 1 (SCICTL1) Field Descriptions

Bit	Field	Value	Description
7	Reserved		Reads return zero; writes have no effect.
6	RX ERR INT ENA		SCI receive error interrupt enable. Setting this bit enables an interrupt if the RX ERROR bit (SCIRXST, bit 7) becomes set because of errors occurring.
		0	Receive error interrupt disabled
		1	Receive error interrupt enabled
5	SW RESET		SCI software reset (active low). Writing a 0 to this bit initializes the SCI state machines and operating flags (registers SCICTL2 and SCIRXST) to the reset condition. The SW RESET bit does not affect any of the configuration bits. All affected logic is held in the specified reset state until a 1 is written to SW RESET (the bit values following a reset are shown beneath each register diagram in this section). Thus, after a system reset, re-enable the SCI by writing a 1 to this bit. Clear this bit after a receiver break detect (BRKDT flag, bit SCIRXST, bit 5). SW RESET affects the operating flags of the SCI, but it neither affects the configuration bits nor restores the reset values. Once SW RESET is asserted, the flags are frozen until the bit is deasserted. The affected flags are as follows:
			Value After SW RESET / SCI Flag / Register Bit 1 / TXRDY / SCICTL2, bit 7 1 / TX EMPTY / SCICTL2, bit 6 0 / RXWAKE / SCIRXST, bit 1 0 / PE / SCIRXST, bit 2 0 / OE / SCIRXST, bit 3 0 / FE / SCIRXST, bit 4 0 / BRKDT / SCIRXST, bit 5 0 / RXRDY / SCIRXST, bit 6 0 / RX ERROR / SCIRXST, bit 7
		0	Writing a 0 to this bit initializes the SCI state machines and operating flags (registers SCICTL2 and SCIRXST) to the reset condition.
		1	After a system reset, re-enable the SCI by writing a 1 to this bit.
4	Reserved		Reads return zero; writes have no effect.
3	TXWAKE		SCI transmitter wake-up method select. The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle-line or address-bit) is specified at the ADDR/IDLE MODE bit (SCICCR, bit 3)
		0	Transmit feature is not selected. In idle-line mode: write a 1 to TXWAKE, then write data to register SCITXBUF to generate an idle period of 11 data bits In address-bit mode: write a 1 to TXWAKE, then write data to SCITXBUF to set the address bit for that frame to 1
		1	Transmit feature selected is dependent on the mode, idle-line or address-bit: TXWAKE is not cleared by the SW RESET bit (SCICTL1, bit 5); it is cleared by a system reset or the transfer of TXWAKE to the WUT flag.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 13-10. SCI Control Register 1 (SCICTL1) Field Descriptions (continued)

Bit	Field	Value	Description
2	SLEEP		SCI sleep. The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle-line or address-bit) is specified at the ADDR/IDLE MODE bit (SCICCR, bit 3). In a multiprocessor configuration, this bit controls the receiver sleep function. Clearing this bit brings the SCI out of the sleep mode. The receiver still operates when the SLEEP bit is set; however, operation does not update the receiver buffer ready bit (SCIRXST, bit 6, RXRDY) or the error status bits (SCIRXST, bit 5−2: BRKDT, FE, OE, and PE) unless the address byte is detected. SLEEP is not cleared when the address byte is detected.
		0	Sleep mode disabled
		1	Sleep mode enabled
1	TXENA		SCI transmitter enable. Data is transmitted through the SCITXD pin only when TXENA is set. If reset, transmission is halted but only after all data previously written to SCITXBUF has been sent.
		0	Transmitter disabled0
		1	Transmitter enabled
0	RXENA		SCI receiver enable. Data is received on the SCIRXD pin and is sent to the receiver shift register and then the receiver buffers. This bit enables or disables the receiver (transfer to the buffers). Clearing RXENA stops received characters from being transferred to the two receiver buffers and also stops the generation of receiver interrupts. However, the receiver shift register can continue to assemble characters. Thus, if RXENA is set during the reception of a character, the complete character will be transferred into the receiver buffer registers, SCIRXEMU and SCIRXBUF.
		0	Prevent received characters from transfer into the SCIRXEMU and SCIRXBUF receiver buffers
		1	Send received characters to SCIRXEMU and SCIRXBUF

Copyright © 2012–2019, Texas Instruments Incorporated

13.3.4 SCI Baud-Select Registers (SCIHBAUD, SCILBAUD)

The values in SCIHBAUD and SCILBAUD specify the baud rate for the SCI.

Figure 13-14. Baud-Select MSbyte Register (SCIHBAUD) — Address 7052h

15	14	13	12	11	10	9	8
BAUD15 (MSB)	BAUD14	BAUD13	BAUD12	BAUD11	BAUD10	BAUD9	BAUD8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 13-15. Baud-Select LSbyte Register (SCILBAUD) — Address 7053h

7	6	5	4	3	2	1	0
BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD10 (LSB)
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-11. Baud-Select Register Field Descriptions

Bit	Field	Value	Description
15-0	BAUD15− BAUD0		SCI 16-bit baud selection Registers SCIHBAUD (MSbyte) and SCILBAUD (LSbyte) are concatenated to form a 16-bit baud value, BRR. The internally-generated serial clock is determined by the low speed peripheral clock (LSPCLK) signal and the two baud-select registers. The SCI uses the 16-bit value of these registers to select one of 64K serial clock rates for the communication modes. The SCI baud rate is calculated using the following equation: $\text{SCI Asynchronous Baud} = \frac{\text{LSPCLK}}{(\text{BRR} + 1) \times 8}$ (8) Alternatively, $\text{BRR} = \frac{\text{LSPCLK}}{\text{SCI Asynchronous Baud} \times 8} - 1$ (9) Note that the above formulas are applicable only when 1 ≤ BRR ≤ 65535. If BRR = 0, then $\text{SCI Asynchronous Baud} = \frac{\text{LSPCLK}}{16}$ (10) Where: BRR = the 16-bit value (in decimal) in the baud-select registers.

Copyright © 2012–2019, Texas Instruments Incorporated

13.3.5 SCI Control Register 2 (SCICTL2)

SCICTL2 enables the receive-ready, break-detect, and transmit-ready interrupts as well as transmitter-ready and -empty flags.

Figure 13-16. SCI Control Register 2 (SCICTL2) — Address 7054h

7	6	5 2	1	0
TXRDY	TX EMPTY	Reserved	RX/BK INT ENA	TX INT ENA
R-1	R-1	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-12. SCI Control Register 2 (SCICTL2) Field Descriptions

Bit	Field	Value	Description
7	TXRDY		Transmitter buffer register ready flag. When set, this bit indicates that the transmit data buffer register, SCITXBUF, is ready to receive another character. Writing data to the SCITXBUF automatically clears this bit. When set, this flag asserts a transmitter interrupt request if the interrupt-enable bit, TX INT ENA (SCICTL2.0), is also set. TXRDY is set to 1 by enabling the SW RESET bit (SCICTL1.5) or by a system reset.
		0	SCITXBUF is full
		1	SCITXBUF is ready to receive the next character
6	TX EMPTY		Transmitter empty flag. This flag's value indicates the contents of the transmitter's buffer register (SCITXBUF) and shift register (TXSHF). An active SW RESET (SCICTL1.5), or a system reset, sets this bit. This bit does not cause an interrupt request.
		0	Transmitter buffer or shift register or both are loaded with data
		1	Transmitter buffer and shift registers are both empty
5-2	Reserved		
1	RX/BK INT ENA		Receiver-buffer/break interrupt enable. This bit controls the interrupt request caused by either the RXRDY flag or the BRKDT flag (bits SCIRXST.6 and .5) being set. However, RX/BK INT ENA does not prevent the setting of these flags.
		0	Disable RXRDY/BRKDT interrupt
		1	Enable RXRDY/BRKDT interrupt
0	TX INT ENA		SCITXBUF-register interrupt enable. This bit controls the interrupt request caused by setting the TXRDY flag bit (SCICTL2.7). However, it does not prevent the TXRDY flag from being set, which indicates SCITXBUF is ready to receive another character.
		0	Disable TXRDY interrupt
		1	Enable TXRDY interrupt
			In non-FIFO mode, a dummy (or a valid) data has to be written to SCITXBUF for the first transmit interrupt to occur. This is the case when you enable the transmit interrupt for the first time and also when you re-enable (disable and then enable) the transmit interrupt. If TXINTENA is enabled after writing the data to SCITXBUF, it will not generate an interrupt.

13.3.6 SCI Receiver Status Register (SCIRXST)

SCIRXST contains seven bits that are receiver status flags (two of which can generate interrupt requests). Each time a complete character is transferred to the receiver buffers (SCIRXEMU and SCIRXBUF), the status flags are updated. Figure 13-18 shows the relationships between several of the register's bits.

Figure 13-17. SCI Receiver Status Register (SCIRXST) — Address 7055h

7	6	5	4	3	2	1	0
RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	Reserved
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

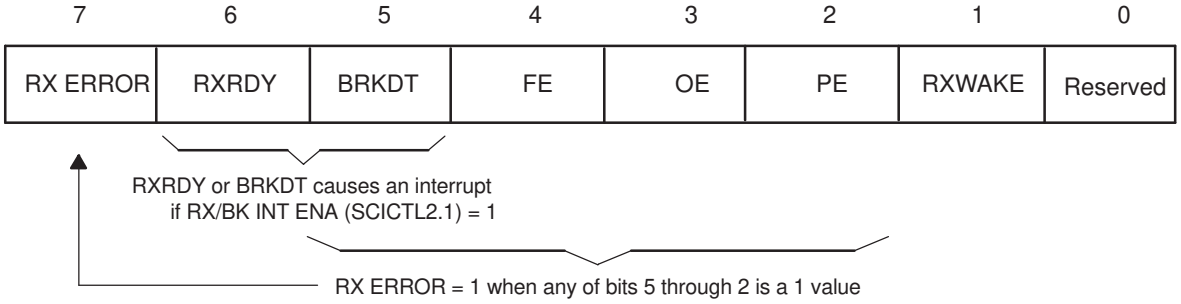
Copyright © 2012–2019, Texas Instruments Incorporated

Table 13-13. SCI Receiver Status Register (SCIRXST) Field Descriptions

Bit	Field	Value	Description
7	RX ERROR		SCI receiver error flag. The RX ERROR flag indicates that one of the error flags in the receiver status register is set. RX ERROR is a logical OR of the break detect, framing error, overrun, and parity error enable flags (bits 5−2: BRKDT, FE, OE, and PE). A 1 on this bit will cause an interrupt if the RX ERR INT ENA bit (SCICTL1.6) is set. This bit can be used for fast error-condition checking during the interrupt service routine. This error flag cannot be cleared directly; it is cleared by an active SW RESET or by a system reset.
		0	No error flags set
		1	Error flag(s) set
6	RXRDY		SCI receiver-ready flag. When a new character is ready to be read from the SCIRXBUF register, the receiver sets this bit, and a receiver interrupt is generated if the RX/BK INT ENA bit (SCICTL2.1) is a 1. RXRDY is cleared by a reading of the SCIRXBUF register, by an active SW RESET, or by a system reset.
		0	No new character in SCIRXBUF
		1	Character ready to be read from SCIRXBUF
5	BRKDT		SCI break-detect flag. The SCI sets this bit when a break condition occurs. A break condition occurs when the SCI receiver data line (SCIRXD) remains continuously low for at least ten bits, beginning after a missing first stop bit. The occurrence of a break causes a receiver interrupt to be generated if the RX/BK INT ENA bit is a 1, but it does not cause the receiver buffer to be loaded. A BRKDT interrupt can occur even if the receiver SLEEP bit is set to 1. BRKDT is cleared by an active SW RESET or by a system reset. It is not cleared by receipt of a character after the break is detected. In order to receive more characters, the SCI must be reset by toggling the SW RESET bit or by a system reset.
		0	No break condition
		1	Break condition occurred
4	FE		SCI framing-error flag. The SCI sets this bit when an expected stop bit is not found. Only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. The FE bit is reset by a clearing of the SW RESET bit or by a system reset.
		0	No framing error detected
		1	Framing error detected
3	OE		SCI overrun-error flag. The SCI sets this bit when a character is transferred into registers SCIRXEMU and SCIRXBUF before the previous character is fully read by the CPU or DMAC. The previous character is overwritten and lost. The OE flag bit is reset by an active SW RESET or by a system reset.
		0	No overrun error detected
		1	Overrun error detected
2	PE		SCI parity-error flag. This flag bit is set when a character is received with a mismatch between the number of 1s and its parity bit. The address bit is included in the calculation. If parity generation and detection is not enabled, the PE flag is disabled and read as 0. The PE bit is reset by an active SW RESET or a system reset.!
		0	No parity error or parity is disabled
		1	Parity error is detected
1	RXWAKE		Receiver wake-up-detect flag
		0	No detection of a receiver wake-up condition
		1	A value of 1 in this bit indicates detection of a receiver wake-up condition. In the address-bit multiprocessor mode (SCICCR.3 = 1), RXWAKE reflects the value of the address bit for the character contained in SCIRXBUF. In the idle-line multiprocessor mode, RXWAKE is set if the SCIRXD data line is detected as idle. RXWAKE is a read-only flag, cleared by one of the following: • The transfer of the first byte after the address byte to SCIRXBUF (only in non-FIFO mode) • The reading of SCIRXBUF • An active SW RESET • A system reset
0	Reserved		Reads return zero; writes have no effect.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 13-18. Register SCIRXST Bit Associations — Address 7055h



13.3.7 Receiver Data Buffer Registers (SCIRXEMU, SCIRXBUF)

Received data is transferred from RXSHF to SCIRXEMU and SCIRXBUF. When the transfer is complete, the RXRDY flag (bit SCIRXST.6) is set, indicating that the received data is ready to be read. Both registers contain the same data; they have separate addresses but are not physically separate buffers. The only difference is that reading SCIRXEMU does not clear the RXRDY flag; however, reading SCIRXBUF clears the flag.

13.3.7.1 Emulation Data Buffer (SCIRXEMU)

Normal SCI data-receive operations read the data received from the SCIRXBUF register. The SCIRXEMU register is used principally by the emulator (EMU) because it can continuously read the data received for screen updates without clearing the RXRDY flag. SCIRXEMU is cleared by a system reset.

This is the register that should be used in an emulator watch window to view the contents of the SCIRXBUF register.

SCIRXEMU is not physically implemented; it is just a different address location to access the SCIRXBUF register without clearing the RXRDY flag.

Figure 13-19. Emulation Data Buffer Register (SCIRXEMU) — Address 7056h

7	6	5	4	3	2	1	0
ERXDT7	ERXDT6	ERXDT5	ERXDT4	ERXDT3	ERXDT2	ERXDT1	ERXDT0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

13.3.7.2 Receiver Data Buffer (SCIRXBUF)

When the current data received is shifted from RXSHF to the receiver buffer, flag bit RXRDY is set and the data is ready to be read. If the RX/BK INT ENA bit (SCICTL2.1) is set, this shift also causes an interrupt. When SCIRXBUF is read, the RXRDY flag is reset. SCIRXBUF is cleared by a system reset.

Figure 13-20. SCI Receive Data Buffer Register (SCIRXBUF) — Address 7057h

15	14	13	12	11	10	9	8
SCIFFFE[1]	SCIFFPE[1]	Reserved					
R−0	R−0	R−0					

7	6	5	4	3	2	1	0
RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0
R−0	R−0	R−0	R−0	R−0	R−0	R−0	R−0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

(1) Applicable only if the FIFO is enabled.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 13-14. SCI Receive Data Buffer Register (SCIRXBUF) Field Descriptions

Bit	Field	Value	Description
15	SCIFFFE		SCIFFFE. SCI FIFO Framing error flag bit (applicable only if the FIFO is enabled)
		0	No frame error occurred while receiving the character, in bits 7−0. This bit is associated with the character on the top of the FIFO.
		1	A frame error occurred while receiving the character in bits 7−0. This bit is associated with the character on the top of the FIFO.
14	SCIFFPE		SCIFFPE. SCI FIFO parity error flag bit (applicable only if the FIFO is enabled)
		0	No parity error occurred while receiving the character, in bits 7−0. This bit is associated with the character on the top of the FIFO.
		1	A parity error occurred while receiving the character in bits 7−0. This bit is associated with the character on the top of the FIFO.
13-8	Reserved		
7-0	RXDT7−0		Receive Character bits

13.3.8 SCI Transmit Data Buffer Register (SCITXBUF)

Data bits to be transmitted are written to SCITXBUF. These bits must be rightjustified because the leftmost bits are ignored for characters less than eight bits long. The transfer of data from this register to the TXSHF transmitter shift register sets the TXRDY flag (SCICTL2.7), indicating that SCITXBUF is ready to receive another set of data. If bit TX INT ENA (SCICTL2.0) is set, this data transfer also causes an interrupt.

Figure 13-21. Transmit Data Buffer Register (SCITXBUF) — Address 7059h

7	6	5	4	3	2	1	0
TXDT7	TXDT6	TXDT5	TXDT4	TXDT3	TXDT2	TXDT1	TXDT0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

13.3.9 SCI FIFO Registers (SCIFFTX, SCIFFRX, SCIFFCT)

The SCI FIFO Registers (SCIFFTX, SCIFFRX, SCIFFCT) are shown and described here.

Figure 13-22. SCI FIFO Transmit (SCIFFTX) Register — Address 705Ah

15	14	13	12	11	10	9	8
SCIRST	SCIFFENA	TXFIFO Reset	TXFFST4	TXFFST3	TXFFST2	TXFFST1	TXFFST0
R/W-1	R/W-0	R/W-1	R−0	R−0	R−0	R−0	R−0
7	6	5	4	3	2	1	0
TXFFINT Flag	TXFFINT CLR	TXFFIENA	TXFFIL4	TXFFIL3	TXFFIL2	TXFFIL1	TXFFIL0
R−0	W−0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-15. SCI FIFO Transmit (SCIFFTX) Register Field Descriptions

Bit	Field	Value	Description
15	SCIRST		SCI Reset
		0	Write 0 to reset the SCI transmit and receive channels. SCI FIFO register configuration bits will be left as is.
		1	SCI FIFO can resume transmit or receive. SCIRST should be 1 even for Autobaud logic to work.
14	SCIFFENA		SCI FIFO enable
		0	SCI FIFO enhancements are disabled
		1	SCI FIFO enhancements are enabled

Copyright © 2012–2019, Texas Instruments Incorporated

Table 13-15. SCI FIFO Transmit (SCIFFTX) Register Field Descriptions (continued)

Bit	Field	Value	Description
13	TXFIFO Reset		Transmit FIFO reset
		0	Reset the FIFO pointer to zero and hold in reset
		1	Re-enable transmit FIFO operation
12-8	TXFFST4-0	00000	Transmit FIFO is empty.
		00001	Transmit FIFO has 1 words
		00010	Transmit FIFO has 2 words
		00011	Transmit FIFO has 3 words
		0xxxx	Transmit FIFO has x words
		10000	Transmit FIFO has 16 words
7	TXFFINT Flag		Transmit FIFO interrupt
		0	TXFIFO interrupt has not occurred, read-only bit
		1	TXFIFO interrupt has occurred, read-only bit
6	TXFFINT CLR		Transmit FIFO clear
		0	Write 0 has no effect on TXFIFINT flag bit, Bit reads back a zero
		1	Write 1 to clear TXFFINT flag in bit 7
5	TXFFIENA		Transmit FIFO interrupt enable
		0	TX FIFO interrupt based on TXFFIL match (less than or equal to) is disabled
		1	TX FIFO interrupt based on TXFFIL match (less than or equal to) is enabled.
4-0	TXFFIL4-0		TXFFIL4−0 Transmit FIFO interrupt level bits. The transmit FIFO generates an interrupt whenever the FIFO status bits (TXFFST4-0) are less than or equal to the FIFO level bits (TXFFIL4-0). The maximum value that can be assigned to these bits to generate an interrupt cannot be more than the depth of the TX FIFO. The default value of these bits after reset is 00000b. Users should set TXFFIL to best fit their application needs by weighing between the CPU overhead to service the ISR and the best possible usage of SCI bus bandwidth.

Figure 13-23. SCI FIFO Receive (SCIFFRX) Register — Address 705Bh

15	14	13	12	11	10	9	8
RXFFOVF	RXFFOVR CLR	RXFIFO Reset	RXFIFST4	RXFFST3	RXFFST2	RXFFST1	RXFFST0
R-0	W−0	R/W−1	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RXFFINT Flag	RXFFINT CLR	RXFFIENA	RXFFIL4	RXFFIL3	RXFFIL2	RXFFIL1	RXFFIL0
R-0	W−0	R/W−0	R/W−1	R/W−1	R/W−1	R/W−1	R/W−1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-16. SCI FIFO Receive (SCIFFRX) Register Field Descriptions

Bit	Field	Value	Description
15	RXFFOVF		Receive FIFO overflow. This will function as flag, but cannot generate interrupt by itself. This condition will occur while receive interrupt is active. Receive interrupts should service this flag condition.
		0	Receive FIFO has not overflowed, read-only bit
		1	Receive FIFO has overflowed, read-only bit. More than 16 words have been received in to the FIFO, and the first received word is lost
14	RXFFOVR CLR		RXFFOVF clear
		0	Write 0 has no effect on RXFFOVF flag bit. Bit reads back a zero
		1	Write 1 to clear RXFFOVF flag in bit 15
13	RXFIFO Reset		Receive FIFO reset
		0	Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1	Re-enable receive FIFO operation

Copyright © 2012–2019, Texas Instruments Incorporated

Table 13-16. SCI FIFO Receive (SCIFFRX) Register Field Descriptions (continued)

Bit	Field	Value	Description
12-8	RXFFST4−0	00000	Receive FIFO is empty
		00001	Receive FIFO has 1 word
		00010	Receive FIFO has 2 words
		00011	Receive FIFO has 3 words
		0xxxx	Receive FIFO has x words
		10000	Receive FIFO has 16 words
7	RXFFINT		Receive FIFO interrupt
		0	RXFIFO interrupt has not occurred, read-only bit
		1	RXFIFO interrupt has occurred, read-only bit
6	RXFFINT CLR		Receive FIFO interrupt clear
		0	Write 0 has no effect on RXFIFINT flag bit. Bit reads back a zero.
		1	Write 1 to clear RXFFINT flag in bit 7
5	RXFFIENA		Receive FIFO interrupt enable
		0	RX FIFO interrupt based on RXFFIL match (greater than or equal to) is disabled
		1	RX FIFO interrupt based on RXFFIL match (less than or equal to) will be enabled.
4-0	RXFFIL4−0		Receive FIFO interrupt level bits
		11111	The receive FIFO generates an interrupt whenever the FIFO status bits (RXFFST4-0) are greater than or equal to the FIFO level bits (RXFFIL4-0). The maximum value that can be assigned to these bits to generate an interrupt cannot be more than the depth of the RX FIFO. The default value of these bits after reset is 11111b. Users should set RXFFIL to best fit their application needs by weighing between the CPU overhead to service the ISR and the best possible usage of received SCI data.

Figure 13-24. SCI FIFO Control (SCIFFCT) Register — Address 705Ch

15	14	13	12				8
ABD	ABD CLR	CDC	Reserved				
R-0	W−0	R/W−0	R-0				

7	6	5	4	3	2	1	0
FFTXDLY7	FFTXDLY6	FFTXDLY5	FFTXDLY4	FFTXDLY3	FFTXDLY2	FFTXDLY1	FFTXDLY0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-17. SCI FIFO Control (SCIFFCT) Register Field Descriptions

Bit	Field	Value	Description
15	ABD		Auto-baud detect (ABD) bit.
		0	Auto-baud detection is not complete. ”A”,”a” character has not been received successfully.
		1	Auto-baud hardware has detected ”A” or ”a” character on the SCI receive register. Auto-detect is complete.
14	ABD CLR		ABD-clear bit
		0	Write 0 has no effect on ABD flag bit. Bit reads back a zero.
		1	Write 1 to clear ABD flag in bit 15.
13	CDC		CDC calibrate A-detect bit
		0	Disables auto-baud alignment
		1	Enables auto-baud alignment
12-8	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 13-17. SCI FIFO Control (SCIFFCT) Register Field Descriptions (continued)

Bit	Field	Value	Description
7−0	FFTXDLY7−0		FIFO transfer delay. These bits define the delay between every transfer from FIFO transmit buffer to transmit shift register. The delay is defined in the number of SCI serial baud clock cycles. The 8 bit register could define a minimum delay of 0 baud clock cycles and a maximum of 256 baud clock cycles In FIFO mode, the buffer (TXBUF) between the shift register and the FIFO should be filled only after the shift register has completed shifting of the last bit. This is required to pass on the delay between transfers to the data stream. In FIFO mode, TXBUF should not be treated as one additional level of buffer. The delayed transmit feature will help to create an auto-flow scheme without RTS/CTS controls as in standard UARTS. When SCI is configured for one stop-bit, delay introduced by FFTXDLY between one frame and the next frame is equal to number of baud clock cycles that FFTXDLY is set to. When SCI is configured for two stop-bits, delay introduced by FFTXDLY between one frame and the next frame is equal to number of baud clock cycles that FFTXDLY is set to minus 1.

Copyright © 2012–2019, Texas Instruments Incorporated

13.3.10 Priority Control Register (SCIPRI)

Figure 13-25. SCI Priority Control Register (SCIPRI) — Address 705Fh

7	5	4	3	2	0
Reserved		SCI SOFT	SCI FREE	Reserved	
R-0		R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-18. SCI Priority Control Register (SCIPRI) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved		Reads return zero; writes have no effect.
4-3	SOFT and FREE		These bits determine what occurs when an emulation suspend event occurs (for example, when the debugger hits a breakpoint). The peripheral can continue whatever it is doing (free-run mode), or if in stop mode, it can either stop immediately or stop when the current operation (the current receive/transmit sequence) is complete.
		00	Immediate stop on suspend
		10	Complete current receive/transmit sequence before stopping
		x1	Free run. Continues SCI operation regardless of suspend
2-0	Reserved		Reads return zero; writes have no effect.

Copyright © 2012–2019, Texas Instruments Incorporated

C28 Inter-Integrated Circuit Module

This chapter describes the features and operation of the inter-integrated circuit (I2C) module. The I2C module provides an interface between one of these devices and devices compliant with Philips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1 and connected by way of an I2C-bus. External components attached to this 2-wire serial bus can transmit/receive 1 to 8-bit data to/from the device through the I2C module. This guide assumes the reader is familiar with the I2C-bus specification.

NOTE: A unit of data transmitted or received by the I2C module can have fewer than 8 bits; however, for convenience, a unit of data is called a data byte throughout this document. The number of bits in a data byte is selectable via the BC bits of the mode register.

Topic **Page**

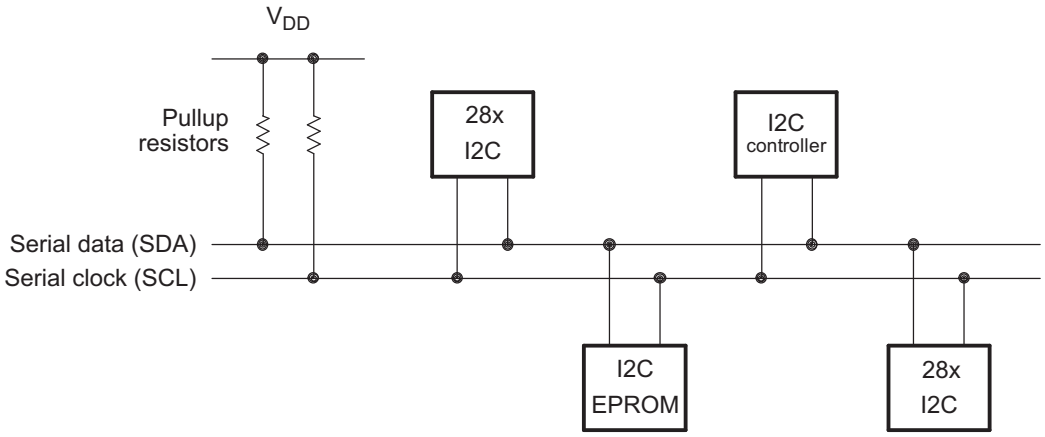
14.1 Introduction to the I2C Module 1008
14.2 I2C Module Operational Details 1011
14.3 Interrupt Requests Generated by the I2C Module 1017
14.4 Resetting/Disabling the I2C Module 1018
14.5 I2C Module Registers 1019

Copyright © 2012–2019, Texas Instruments Incorporated

14.1 Introduction to the I2C Module

The I2C module supports any slave or master I2C-compatible device. Figure 14-1 shows an example of multiple I2C modules connected for a two-way transfer from one device to other devices.

Figure 14-1. Multiple I2C Modules Connected



Copyright © 2012–2019, Texas Instruments Incorporated

14.1.1 Features

The I2C module has the following features:

- Compliance with the Philips Semiconductors I2C-bus specification (version 2.1):
  - Support for 8-bit format transfers
  - 7-bit and 10-bit addressing modes
  - General call
  - START byte mode
  - Support for multiple master-transmitters and slave-receivers
  - Support for multiple slave-transmitters and master-receivers
  - Combined master transmit/receive and receive/transmit mode
  - Data transfer rate of from 10 kbps up to 400 kbps (Philips Fast-mode rate)
- One 16-byte receive FIFO and one 16-byte transmit FIFO
- One interrupt that can always be used by the CPU. This interrupt can be generated as a result of one of the following conditions: transmit-data ready, receive-data ready, register-access ready, no-acknowledgment received, arbitration lost, stop condition detected, addressed as slave.
- An additional interrupt that can be used by the CPU when in FIFO mode
- Module enable/disable capability
- Free data format mode

14.1.2 Features Not Supported

The I2C module does not support:

- High-speed mode (Hs-mode)
- CBUS-compatibility mode

14.1.3 Functional Overview

Each device connected to an I2C-bus is recognized by a unique address. Each device can operate as either a transmitter or a receiver, depending on the function of the device. A device connected to the I2C-bus can also be considered as the master or the slave when performing data transfers. A master device is the device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. During this transfer, any device addressed by this master is considered a slave. The I2C module supports the multi-master mode, in which one or more devices capable of controlling an I2C-bus can be connected to the same I2C-bus.

For data communication, the I2C module has a serial data pin (SDA) and a serial clock pin (SCL), as shown in Figure 14-2. These two pins carry information between the 28x device and other devices connected to the I2C-bus. The SDA and SCL pins both are bidirectional. They each must be connected to a positive supply voltage using a pull-up resistor. When the bus is free, both pins are high. The driver of these two pins has an open-drain configuration to perform the required wired-AND function.

There are two major transfer techniques:

- Standard Mode: Send exactly n data values, where n is a value you program in an I2C module register. See Table 14-5 for register information.
- Repeat Mode: Keep sending data values until you use software to initiate a STOP condition or a new START condition. See Table 14-5 for RM bit information.

.

The I2C module consists of the following primary blocks:

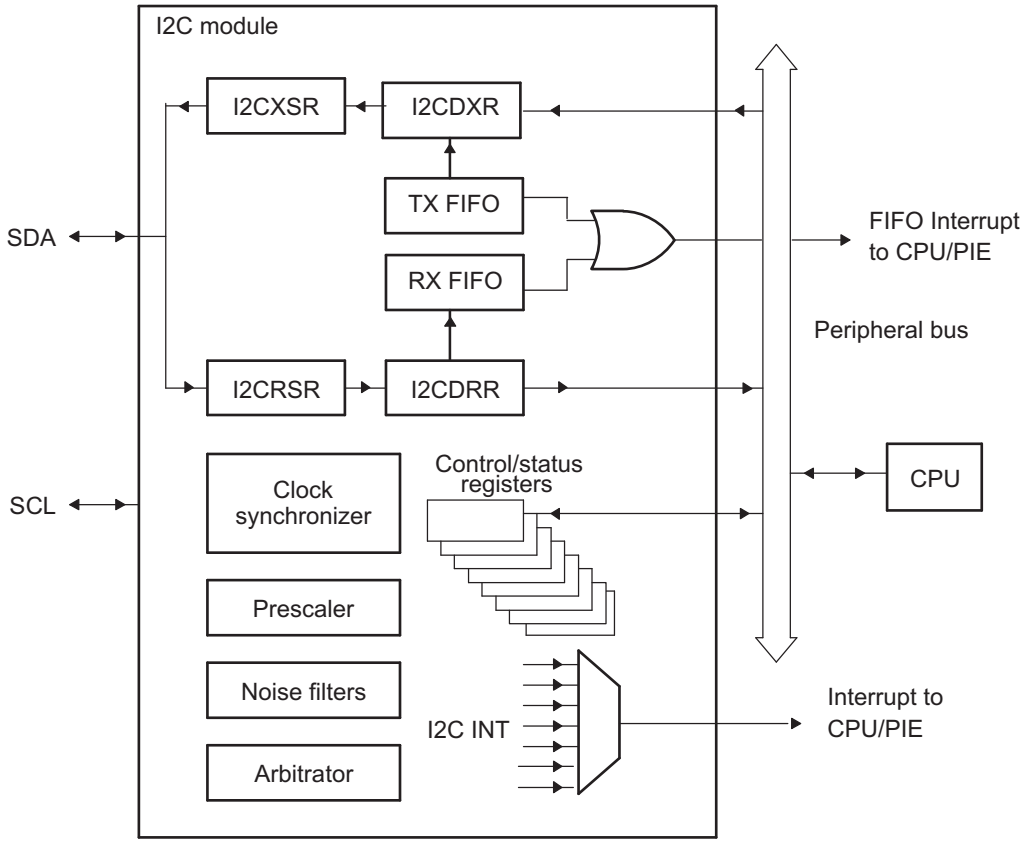
- A serial interface: one data pin (SDA) and one clock pin (SCL)
- Data registers and FIFOs to temporarily hold receive data and transmit data traveling between the SDA pin and the CPU
- Control and status registers
- A peripheral bus interface to enable the CPU to access the I2C module registers and FIFOs.

Copyright © 2012–2019, Texas Instruments Incorporated

- A clock synchronizer to synchronize the I2C input clock (from the device clock generator) and the clock on the SCL pin, and to synchronize data transfers with masters of different clock speeds
- A prescaler to divide down the input clock that is driven to the I2C module
- A noise filter on each of the two pins, SDA and SCL
- An arbitrator to handle arbitration between the I2C module (when it is a master) and another master
- Interrupt generation logic, so that an interrupt can be sent to the CPU
- FIFO interrupt generation logic, so that FIFO access can be synchronized to data reception and data transmission in the I2C module

Figure 14-2 shows the four registers used for transmission and reception in non-FIFO mode. The CPU writes data for transmission to I2CDXR and reads received data from I2CDRR. When the I2C module is configured as a transmitter, data written to I2CDXR is copied to I2CXSR and shifted out on the SDA pin one bit a time. When the I2C module is configured as a receiver, received data is shifted into I2CRSR and then copied to I2CDRR.

Figure 14-2. I2C Module Conceptual Block Diagram

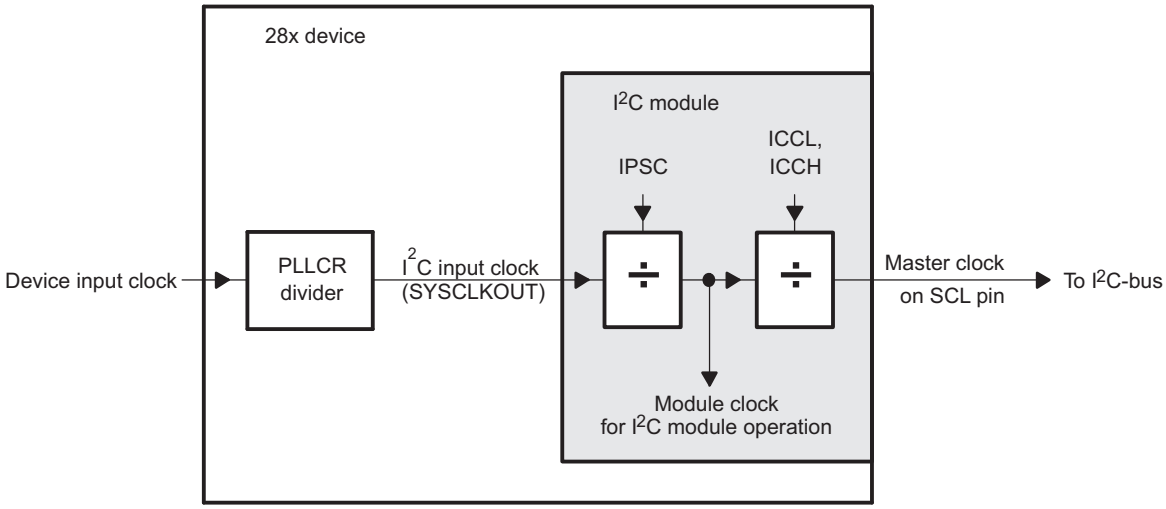


14.1.4 Clock Generation

As shown in Figure 14-3, the device clock generator receives a signal from an external clock source and produces an I2C input clock with a programmed frequency. The I2C input clock is equivalent to the CPU clock and is then divided twice more inside the I2C module to produce the module clock and the master clock.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 14-3. Clocking Diagram for the I2C Module



The module clock determines the frequency at which the I2C module operates. A programmable prescaler in the I2C module divides down the I2C input clock to produce the module clock. To specify the divide-down value, initialize the IPSC field of the prescaler register, I2CPSC. The resulting frequency is:

$$\text{module clock frequency} = \frac{\text{I2C input clock frequency}}{(\text{IPSC} + 1)}$$

NOTE: To meet all of the I2C protocol timing specifications, the module clock must be configured between 7 - 12 MHz.

The prescaler must be initialized only while the I2C module is in the reset state (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

The master clock appears on the SCL pin when the I2C module is configured to be a master on the I2C-bus. This clock controls the timing of communication between the I2C module and a slave. As shown in Figure 14-3, a second clock divider in the I2C module divides down the module clock to produce the master clock. The clock divider uses the ICCL value of I2CCLKL to divide down the low portion of the module clock signal and uses the ICCH value of I2CCLKH to divide down the high portion of the module clock signal. See section Section 14.5.7.1 for the master clock frequency equation.

14.2 I2C Module Operational Details

This section provides an overview of the I2C-bus protocol and how it is implemented.

14.2.1 Input and Output Voltage Levels

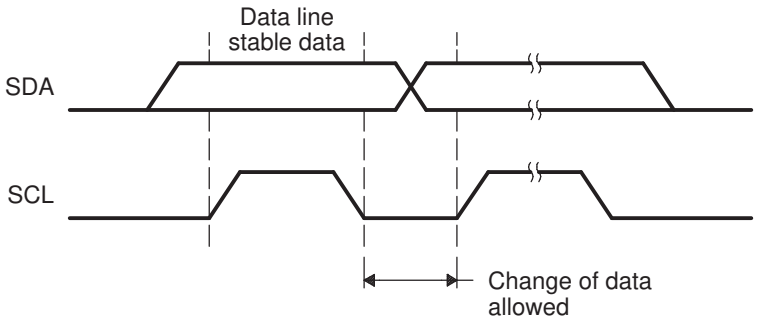
One clock pulse is generated by the master device for each data bit transferred. Due to a variety of different technology devices that can be connected to the I2C-bus, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the associated level of V_{DD}. For details, see the data manual for your particular device.

14.2.2 Data Validity

The data on SDA must be stable during the high period of the clock (see Figure 14-4). The high or low state of the data line, SDA, should change only when the clock signal on SCL is low.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 14-4. Bit Transfer on the I2C-Bus



14.2.3 Operating Modes

The I2C module has four basic operating modes to support data transfers as a master and as a slave. See Table 14-1 for the names and descriptions of the modes.

If the I2C module is a master, it begins as a master-transmitter and typically transmits an address for a particular slave. When giving data to the slave, the I2C module must remain a master-transmitter. To receive data from a slave, the I2C module must be changed to the master-receiver mode.

If the I2C module is a slave, it begins as a slave-receiver and typically sends acknowledgment when it recognizes its slave address from a master. If the master will be sending data to the I2C module, the module must remain a slave-receiver. If the master has requested data from the I2C module, the module must be changed to the slave-transmitter mode.

Table 14-1. Operating Modes of the I2C Module

Operating Mode	Description
Slave-receiver modes	The I2C module is a slave and receives data from a master.
	All slaves begin in this mode. In this mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master. As a slave, the I2C module does not generate the clock signal, but it can hold SCL low while the intervention of the device is required (RSFULL = 1 in I2CSTR) after a byte has been received. See section Section 14.2.7 for more details.
Slave-transmitter mode	The I2C module is a slave and transmits data to a master.
	This mode can be entered only from the slave-receiver mode; the I2C module must first receive a command from the master. When you are using any of the 7-bit/10-bit addressing formats, the I2C module enters its slave-transmitter mode if the slave address byte is the same as its own address (in I2COAR) and the master has transmitted R/$\overline{W}$ = 1. As a slave-transmitter, the I2C module then shifts the serial data out on SDA with the clock pulses that are generated by the master. While a slave, the I2C module does not generate the clock signal, but it can hold SCL low while the intervention of the device is required (XSMT = 0 in I2CSTR) after a byte has been transmitted. See section Section 14.2.7 for more details.
Master-receiver mode	The I2C module is a master and receives data from a slave.
	This mode can be entered only from the master-transmitter mode; the I2C module must first transmit a command to the slave. When you are using any of the 7-bit/10-bit addressing formats, the I2C module enters its master-receiver mode after transmitting the slave address byte and R/$\overline{W}$ = 1. Serial data bits on SDA are shifted into the I2C module with the clock pulses generated by the I2C module on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the device is required (RSFULL = 1 in I2CSTR) after a byte has been received.
Master-transmitter modes	The IC module is a master and transmits control information and data to a slave.
	All masters begin in this mode. In this mode, data assembled in any of the 7-bit/10-bit addressing formats is shifted out on SDA. The bit shifting is synchronized with the clock pulses generated by the I2C module on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the device is required (XSMT = 0 in I2CSTR) after a byte has been transmitted.

Copyright © 2012–2019, Texas Instruments Incorporated

To summarize, SCL will be held low in the following conditions:

- When RSFULL = 1, in Slave-receiver mode.
- When XSMT = 0, in Slave-transmitter mode.

I2C slave nodes have to accept and provide data when the I2C master node requests it.

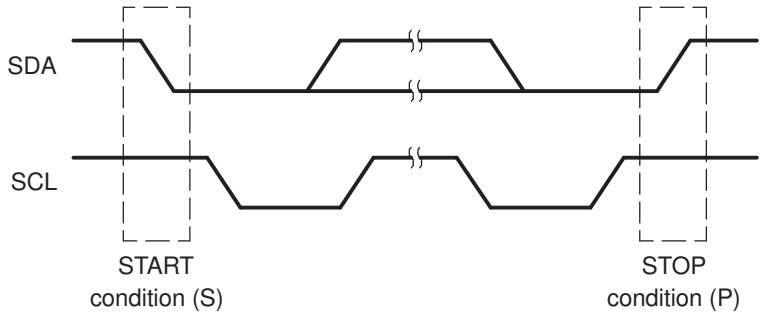
- To release SCL in slave-receiver mode, read data from I2CDRR.
- To release SCL in slave-transmitter mode, write data to I2CDXR.
- To force a release without handling the data, reset the module using the I2CMDR.IRS bit.

14.2.4 I2C Module START and STOP Conditions

START and STOP conditions can be generated by the I2C module when the module is configured to be a master on the I2C-bus. As shown in Figure 14-5:

- The START condition is defined as a high-to-low transition on the SDA line while SCL is high. A master drives this condition to indicate the start of a data transfer.
- The STOP condition is defined as a low-to-high transition on the SDA line while SCL is high. A master drives this condition to indicate the end of a data transfer.

Figure 14-5. I2C Module START and STOP Conditions



After a START condition and before a subsequent STOP condition, the I2C-bus is considered busy, and the bus busy (BB) bit of I2CSTR is 1. Between a STOP condition and the next START condition, the bus is considered free, and BB is 0.

For the I2C module to start a data transfer with a START condition, the master mode bit (MST) and the START condition bit (STT) in I2CMDR must both be 1. For the I2C module to end a data transfer with a STOP condition, the STOP condition bit (STP) must be set to 1. When the BB bit is set to 1 and the STT bit is set to 1, a repeated START condition is generated. For a description of I2CMDR and its bits (including MST, STT, and STP), see Section 14.5.1.

14.2.5 Serial Data Formats

Figure 14-6 shows an example of a data transfer on the I2C-bus. The I2C module supports 1 to 8-bit data values. In Figure 14-6, 8-bit data is transferred. Each bit put on the SDA line equates to 1 pulse on the SCL line, and the values are always transferred with the most significant bit (MSB) first. The number of data values that can be transmitted or received is unrestricted. The serial data format used in Figure 14-6 is the 7-bit addressing format. The I2C module supports the formats shown in Figure 14-7 through Figure 14-9 and described in the paragraphs that follow the figures.

NOTE: In Figure 14-6 through Figure 14-9, n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 14-6. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown)

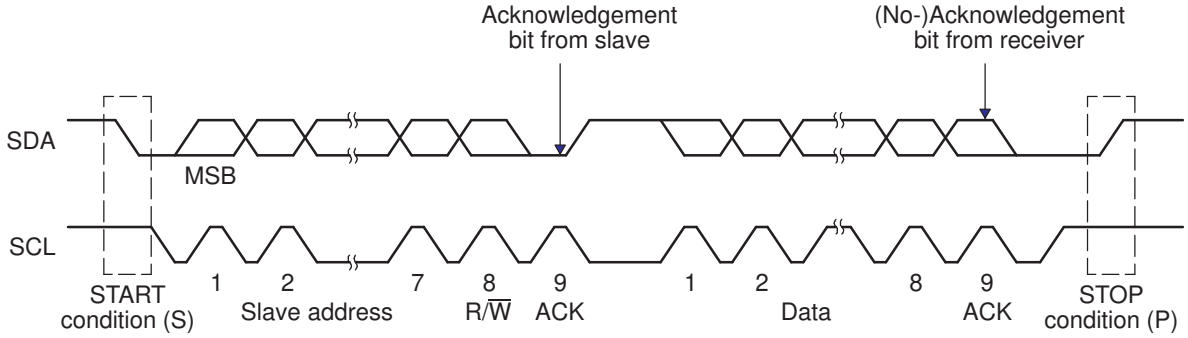


Figure 14-7. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)

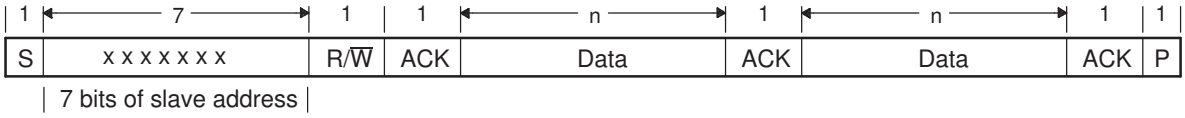


Figure 14-8. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR)

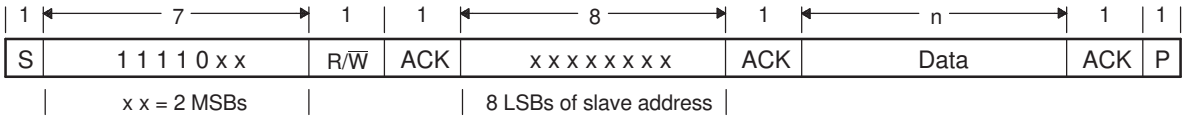
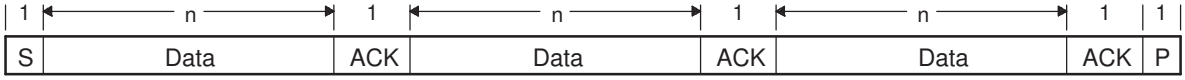


Figure 14-9. I2C Module Free Data Format (FDF = 1 in I2CMDR)



14.2.5.1 7-Bit Addressing Format

In the 7-bit addressing format (see Figure 14-7), the first byte after a START condition (S) consists of a 7-bit slave address followed by a R/$\overline{W}$ bit. R/$\overline{W}$ determines the direction of the data:

- R/$\overline{W}$ = 0: The master writes (transmits) data to the addressed slave.
- R/$\overline{W}$ = 1: The master reads (receives) data from the slave.

An extra clock cycle dedicated for acknowledgment (ACK) is inserted after each byte. If the ACK bit is inserted by the slave after the first byte from the master, it is followed by n bits of data from the transmitter (master or slave, depending on the R/$\overline{W}$ bit). n is a number from 1 to 8 determined by the bit count (BC) field of I2CMDR. After the data bits have been transferred, the receiver inserts an ACK bit.

To select the 7-bit addressing format, write 0 to the expanded address enable (XA) bit of I2CMDR, and make sure the free data format mode is off (FDF = 0 in I2CMDR).

14.2.5.2 10-Bit Addressing Format

The 10-bit addressing format (see Figure 14-8) is similar to the 7-bit addressing format, but the master sends the slave address in two separate byte transfers. The first byte consists of 11110b, the two MSBs of the 10-bit slave address, and R/$\overline{W}$ = 0 (write). The second byte is the remaining 8 bits of the 10-bit slave address. The slave must send acknowledgment after each of the two byte transfers. Once the master has written the second byte to the slave, the master can either write data or use a repeated START condition to change the data direction. For more details about using 10-bit addressing, see the Philips Semiconductors I2C-bus specification.

To select the 10-bit addressing format, write 1 to the XA bit of I2CMDR and make sure the free data format mode is off (FDF = 0 in I2CMDR).

Copyright © 2012–2019, Texas Instruments Incorporated

14.2.5.3 Free Data Format

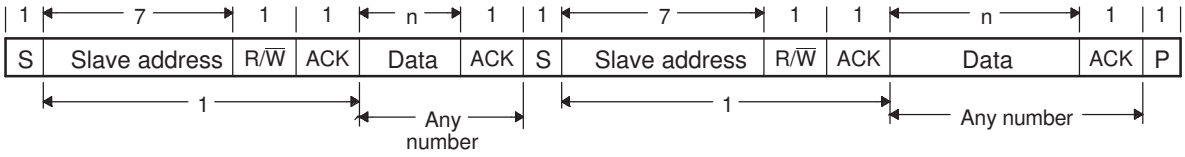
In this format (see Figure 14-9), the first byte after a START condition (S) is a data byte. An ACK bit is inserted after each data byte, which can be from 1 to 8 bits, depending on the BC field of I2CMDR. No address or data-direction bit is sent. Therefore, the transmitter and the receiver must both support the free data format, and the direction of the data must be constant throughout the transfer.

To select the free data format, write 1 to the free data format (FDF) bit of I2CMDR. The free data format is not supported in the digital loopback mode (DLB = 1 in I2CMDR).

14.2.5.4 Using a Repeated START Condition

At the end of each data byte, the master can drive another START condition. Using this capability, a master can communicate with multiple slave addresses without having to give up control of the bus by driving a STOP condition. The length of a data byte can be from 1 to 8 bits and is selected with the BC field of I2CMDR. The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, and free data formats. Figure 14-10 shows a repeated START condition in the 7-bit addressing format.

Figure 14-10. Repeated START Condition (in This Case, 7-Bit Addressing Format)



NOTE: In Figure 14-10, n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

14.2.6 NACK Bit Generation

When the I2C module is a receiver (master or slave), it can acknowledge or ignore bits sent by the transmitter. To ignore any new bits, the I2C module must send a no-acknowledge (NACK) bit during the acknowledge cycle on the bus. Table 14-2 summarizes the various ways you can tell the I2C module to send a NACK bit.

Table 14-2. Ways to Generate a NACK Bit

I2C Module Condition	NACK Bit Generation Options
Slave-receiver modes	• Allow an overrun condition (RSFULL = 1 in I2CSTR) • Reset the module (IRS = 0 in I2CMDR) • Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive
Master-receiver mode AND Repeat mode (RM = 1 in I2CMDR)	• Generate a STOP condition (STP = 1 in I2CMDR) • Reset the module (IRS = 0 in I2CMDR) • Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive
Master-receiver mode AND Nonrepeat mode (RM = 0 in I2CMDR)	• If STP = 1 in I2CMDR, allow the internal data counter to count down to 0 and thus force a STOP condition • If STP = 0, make STP = 1 to generate a STOP condition • Reset the module (IRS = 0 in I2CMDR). = 1 to generate a STOP condition • Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive

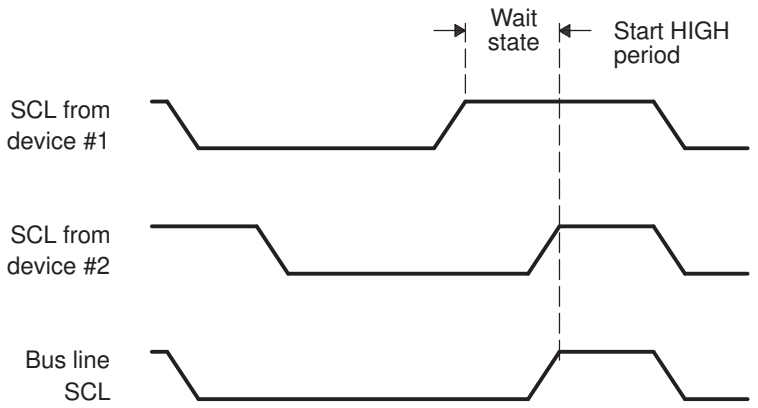
Copyright © 2012–2019, Texas Instruments Incorporated

14.2.7 Clock Synchronization

Under normal conditions, only one master device generates the clock signal, SCL. During the arbitration procedure, however, there are two or more masters and the clock must be synchronized so that the data output can be compared. Figure 14-11 illustrates the clock synchronization. The wired-AND property of SCL means that a device that first generates a low period on SCL overrules the other devices. At this high-to-low transition, the clock generators of the other devices are forced to start their own low period. The SCL is held low by the device with the longest low period. The other devices that finish their low periods must wait for SCL to be released, before starting their high periods. A synchronized signal on SCL is obtained, where the slowest device determines the length of the low period and the fastest device determines the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the wait state. In this way, a slave slows down a fast master and the slow device creates enough time to store a received byte or to prepare a byte to be transmitted.

Figure 14-11. Synchronization of Two I2C Clock Generators During Arbitration



14.2.8 Arbitration

If two or more master-transmitters attempt to start a transmission on the same bus at approximately the same time, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (SDA) by the competing transmitters. Figure 14-12 illustrates the arbitration procedure between two devices. The first master-transmitter, which release the SDA line high, is overruled by another master-transmitter that drives SDA low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

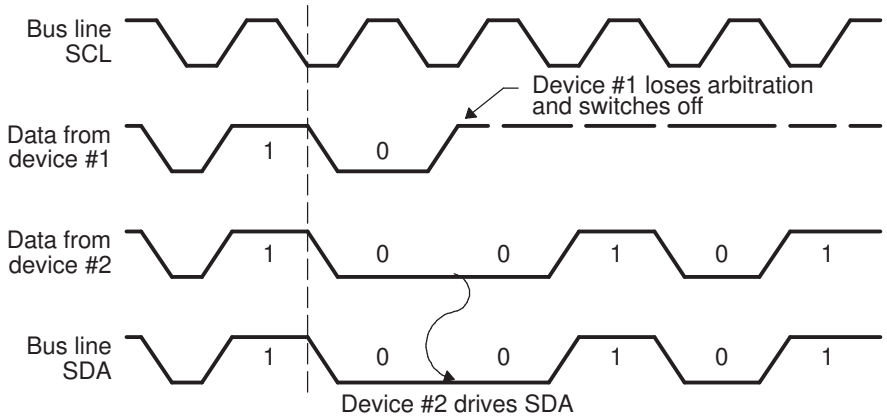
If the I2C module is the losing master, it switches to the slave-receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration-lost interrupt request.

If during a serial transfer the arbitration procedure is still in progress when a repeated START condition or a STOP condition is transmitted to SDA, the master-transmitters involved must send the repeated START condition or the STOP condition at the same position in the format frame. Arbitration is not allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 14-12. Arbitration Procedure Between Two Master-Transmitters



14.3 Interrupt Requests Generated by the I2C Module

The I2C module can generate seven types of basic interrupt requests, which are described in Section 14.3.1. Two of these can tell the CPU when to write transmit data and when to read receive data. If you want the FIFOs to handle transmit and receive data, you can also use the FIFO interrupts described in Section 14.3.2. The basic I2C interrupts are combined to form PIE Group 8, Interrupt 1 (I2CINT1A_ISR), and the FIFO interrupts are combined to form PIE Group 8, Interrupt 2 (I2CINT2A_ISR).

14.3.1 Basic I2C Interrupt Requests

The I2C module generates the interrupt requests described in Table 14-3. As shown in Figure 14-13, all requests are multiplexed through an arbiter to a single I2C interrupt request to the CPU. Each interrupt request has a flag bit in the status register (I2CSTR) and an enable bit in the interrupt enable register (I2CIER). When one of the specified events occurs, its flag bit is set. If the corresponding enable bit is 0, the interrupt request is blocked. If the enable bit is 1, the request is forwarded to the CPU as an I2C interrupt.

The I2C interrupt is one of the maskable interrupts of the CPU. As with any maskable interrupt request, if it is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (I2CINT1A_ISR). The I2CINT1A_ISR for the I2C interrupt can determine the interrupt source by reading the interrupt source register, I2CISRC. Then the I2CINT1A_ISR can branch to the appropriate subroutine.

After the CPU reads I2CISRC, the following events occur:

1. The flag for the source interrupt is cleared in I2CSTR. Exception: The ARDY, RRDY, and XRDY bits in I2CSTR are not cleared when I2CISRC is read. To clear one of these bits, write a 1 to it.
2. The arbiter determines which of the remaining interrupt requests has the highest priority, writes the code for that interrupt to I2CISRC, and forwards the interrupt request to the CPU.

Table 14-3. Descriptions of the Basic I2C Interrupt Requests

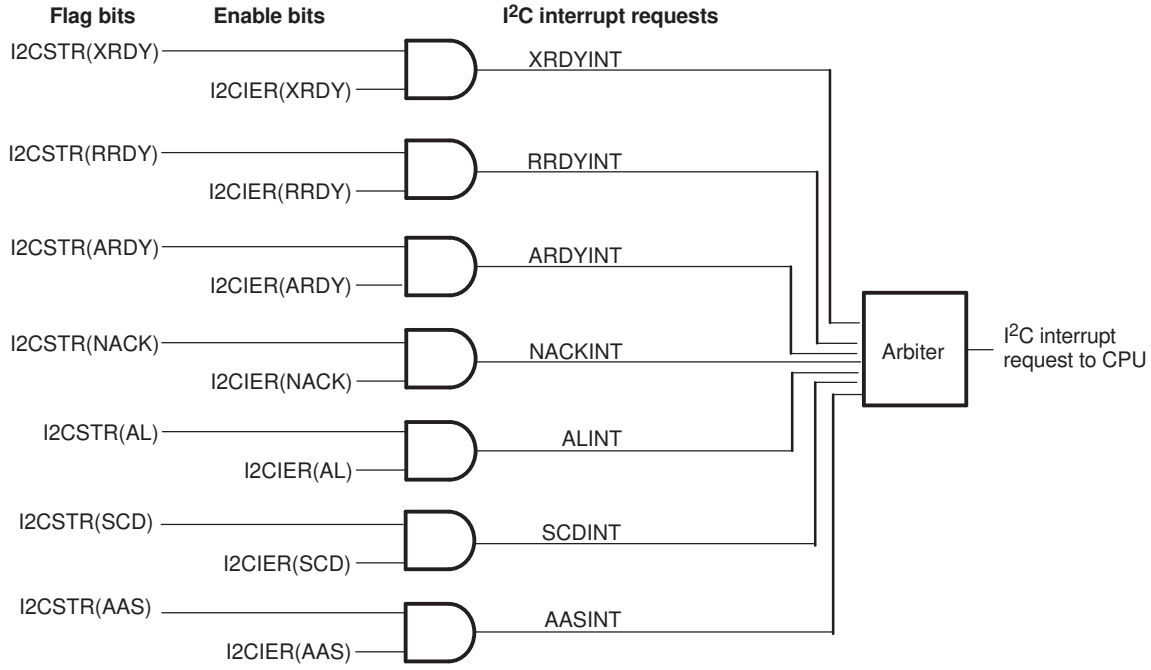
I2C Interrupt Request	Interrupt Source
XRDYINT	Transmit ready condition: The data transmit register (I2CDXR) is ready to accept new data because the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR). As an alternative to using XRDYINT, the CPU can poll the XRDY bit of the status register, I2CSTR. XRDYINT should not be used when in FIFO mode. Use the FIFO interrupts instead.
RRDYINT	Receive ready condition: The data receive register (I2CDRR) is ready to be read because data has been copied from the receive shift register (I2CRSR) to I2CDRR. As an alternative to using RRDYINT, the CPU can poll the RRDY bit of I2CSTR. RRDYINT should not be used when in FIFO mode. Use the FIFO interrupts instead.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 14-3. Descriptions of the Basic I2C Interrupt Requests (continued)

I2C Interrupt Request	Interrupt Source
ARDYINT	Register-access ready condition: The I2C module registers are ready to be accessed because the previously programmed address, data, and command values have been used. The specific events that generate ARDYINT are the same events that set the ARDY bit of I2CSTR. As an alternative to using ARDYINT, the CPU can poll the ARDY bit.
NACKINT	No-acknowledgment condition: The I2C module is configured as a master-transmitter and did not received acknowledgment from the slave-receiver. As an alternative to using NACKINT, the CPU can poll the NACK bit of I2CSTR.
ALINT	Arbitration-lost condition: The I2C module has lost an arbitration contest with another master-transmitter. As an alternative to using ALINT, the CPU can poll the AL bit of I2CSTR.
SCDINT	Stop condition detected: A STOP condition was detected on the I2C bus. As an alternative to using SCDINT, the CPU can poll the SCD bit of the status register, I2CSTR.
AASINT	Addressed as slave condition: The I2C has been addressed as a slave device by another master on the I2C bus. As an alternative to using AASINT, the CPU can poll the AAS bit of the status register, I2CSTR.

Figure 14-13. Enable Paths of the I2C Interrupt Requests



14.3.2 I2C FIFO Interrupts

In addition to the seven basic I2C interrupts, the transmit and receive FIFOs each contain the ability to generate an interrupt (I2CINT2A). The transmit FIFO can be configured to generate an interrupt after transmitting a defined number of bytes, up to 16 . The receive FIFO can be configured to generate an interrupt after receiving a defined number of bytes, up to 16. These two interrupt sources are ORed together into a single maskable CPU interrupt. The interrupt service routine can then read the FIFO interrupt status flags to determine from which source the interrupt came. See the I2C transmit FIFO register (I2CFFTX) and the I2C receive FIFO register (I2CFFRX) descriptions.

14.4 Resetting/Disabling the I2C Module

You can reset/disable the I2C module in two ways:

- Write 0 to the I2C reset bit (IRS) in the I2C mode register (I2CMDR). All status bits (in I2CSTR) are

Copyright © 2012–2019, Texas Instruments Incorporated

forced to their default values, and the I2C module remains disabled until IRS is changed to 1. The SDA and SCL pins are in the high-impedance state.

- Initiate a device reset by driving the $\overline{\text{XRS}}$ pin low. The entire device is reset and is held in the reset state until you drive the pin high. When $\overline{\text{XRS}}$ is released, all I2C module registers are reset to their default values. The IRS bit is forced to 0, which resets the I2C module. The I2C module stays in the reset state until you write 1 to IRS.

IRS must be 0 while you configure/reconfigure the I2C module. Forcing IRS to 0 can be used to save power and to clear error conditions.

14.5 I2C Module Registers

Table 14-4 lists the I2C module registers. All but the receive and transmit shift registers (I2CRSR and I2CXSR) are accessible to the CPU.

Table 14-4. I2C Module Registers

Name	Address	Description
I2COAR	0x7900	I2C own address register
I2CIER	0x7901	I2C interrupt enable register
I2CSTR	0x7902	I2C status register
I2CCLKL	0x7903	I2C clock low-time divider register
I2CCLKH	0x7904	I2C clock high-time divider register
I2CCNT	0x7905	I2C data count register
I2CDRR	0x7906	I2C data receive register
I2CSAR	0x7907	I2C slave address register
I2CDXR	0x7908	I2C data transmit register
I2CMDR	0x7909	I2C mode register
I2CISRC	0x790A	I2C interrupt source register
I2CEMDR	0x790B	I2C extended mode register
I2CPSC	0x790C	I2C prescaler register
I2CFFTX	0x7920	I2C FIFO transmit register
I2CFFRX	0x7921	I2C FIFO receive register
I2CRSR	-	I2C receive shift register (not accessible to the CPU)
I2CXSR	-	I2C transmit shift register (not accessible to the CPU)

NOTE: To use the I2C module the system clock to the module must be enabled by setting the appropriate bit in the PCLKR0 register. See *TMS320x2833x, 2823x device System Control and Interrupts Reference Guide* (literature number SPRUFB0).

Copyright © 2012–2019, Texas Instruments Incorporated

14.5.1 I2C Mode Register (I2CMDR)

The I2C mode register (I2CMDR) is a 16-bit register that contains the control bits of the I2C module. The bit fields of I2CMDR are shown in Figure 14-14 and described in Table 14-5.

Figure 14-14. I2C Mode Register (I2CMDR)

15	14	13	12	11	10	9	8
NACKMOD	FREE	STT	Reserved	STP	MST	TRX	XA
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2		0
RM	DLB	IRS	STB	FDF	BC		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 14-5. I2C Mode Register (I2CMDR) Field Descriptions

Bit	Field	Value	Description
15	NACKMOD		NACK mode bit. This bit is only applicable when the I2C module is acting as a receiver.
		0	In the slave-receiver mode: The I2C module sends an acknowledge (ACK) bit to the transmitter during each acknowledge cycle on the bus. The I2C module only sends a no-acknowledge (NACK) bit if you set the NACKMOD bit. In the master-receiver mode: The I2C module sends an ACK bit during each acknowledge cycle until the internal data counter counts down to 0. At that point, the I2C module sends a NACK bit to the transmitter. To have a NACK bit sent earlier, you must set the NACKMOD bit.
		1	In either slave-receiver or master-receiver mode: The I2C module sends a NACK bit to the transmitter during the next acknowledge cycle on the bus. Once the NACK bit has been sent, NACKMOD is cleared. Important: To send a NACK bit in the next acknowledge cycle, you must set NACKMOD before the rising edge of the last data bit.
14	FREE		This bit controls the action taken by the I2C module when a debugger breakpoint is encountered.
		0	When I2C module is master: If SCL is low when the breakpoint occurs, the I2C module stops immediately and keeps driving SCL low, whether the I2C module is the transmitter or the receiver. If SCL is high, the I2C module waits until SCL becomes low and then stops. When I2C module is slave: A breakpoint forces the I2C module to stop when the current transmission/reception is complete.
		1	The I2C module runs free; that is, it continues to operate when a breakpoint occurs.
13	STT		START condition bit (only applicable when the I2C module is a master). The RM, STT, and STP bits determine when the I2C module starts and stops data transmissions (see Table 14-6). Note that the STT and STP bits can be used to terminate the repeat mode, and that this bit is not writable when IRS = 0.
		0	In the master mode, STT is automatically cleared after the START condition has been generated.
		1	In the master mode, setting STT to 1 causes the I2C module to generate a START condition on the I2C-bus.
12	Reserved		This reserved bit location is always read as a 0. A value written to this bit has no effect.
11	STP		STOP condition bit (only applicable when the I2C module is a master). In the master mode, the RM, STT, and STP bits determine when the I2C module starts and stops data transmissions (see Table 14-6). Note that the STT and STP bits can be used to terminate the repeat mode, and that this bit is not writable when IRS=0. When in non-repeat mode, at least one byte must be transferred before a stop condition can be generated. The I2C module delays clearing of this bit until ater the I2CSTR[SCD] bit is set. If the STOP bit is not checked prior to initiating a new message, the I2C module could become confused. To avoid disrupting the I2C state machine, the user must wait until this bit is clear before initiating a new message.
		0	STP is automatically cleared after the STOP condition has been generated.
		1	STP has been set by the device to generate a STOP condition when the internal data counter of the I2C module counts down to 0.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 14-5. I2C Mode Register (I2CMDR) Field Descriptions (continued)

Bit	Field	Value	Description
10	MST		Master mode bit. MST determines whether the I2C module is in the slave mode or the master mode. MST is automatically changed from 1 to 0 when the I2C master generates a STOP condition.
		0	Slave mode. The I2C module is a slave and receives the serial clock from the master.
		1	Master mode. The I2C module is a master and generates the serial clock on the SCL pin.
9	TRX		Transmitter mode bit. When relevant, TRX selects whether the I2C module is in the transmitter mode or the receiver mode. Table 14-7 summarizes when TRX is used and when it is a don't care.
		0	Receiver mode. The I2C module is a receiver and receives data on the SDA pin.
		1	Transmitter mode. The I2C module is a transmitter and transmits data on the SDA pin.
8	XA		Expanded address enable bit.
		0	7-bit addressing mode (normal address mode). The I2C module transmits 7-bit slave addresses (from bits 6-0 of I2CSAR), and its own slave address has 7 bits (bits 6-0 of I2COAR).
		1	10-bit addressing mode (expanded address mode). The I2C module transmits 10-bit slave addresses (from bits 9-0 of I2CSAR), and its own slave address has 10 bits (bits 9-0 of I2COAR).
7	RM		Repeat mode bit (only applicable when the I2C module is a master-transmitter). The RM, STT, and STP bits determine when the I2C module starts and stops data transmissions (see Table 14-6).
		0	Nonrepeat mode. The value in the data count register (I2CCNT) determines how many bytes are received/transmitted by the I2C module.
		1	Repeat mode. A data byte is transmitted each time the I2CDXR register is written to (or until the transmit FIFO is empty when in FIFO mode) until the STP bit is manually set. The value of I2CCNT is ignored. The ARDY bit/interrupt can be used to determine when the I2CDXR (or FIFO) is ready for more data, or when the data has all been sent and the CPU is allowed to write to the STP bit.
6	DLB		Digital loopback mode bit. The effects of this bit are shown in Figure 14-15.
		0	Digital loopback mode is disabled.
		1	Digital loopback mode is enabled. For proper operation in this mode, the MST bit must be 1.
			In the digital loopback mode, data transmitted out of I2CDXR is received in I2CDRR after n device cycles by an internal path, where:
			n = ((I2C input clock frequency/module clock frequency) × 8)
			The transmit clock is also the receive clock. The address transmitted on the SDA pin is the address in I2COAR.
			Note: The free data format (FDF = 1) is not supported in the digital loopback mode.
5	IRS		I2C module reset bit.
		0	The I2C module is in reset/disabled. When this bit is cleared to 0, all status bits (in I2CSTR) are set to their default values.
		1	The I2C module is enabled. This has the effect of releasing the I2C bus if the I2C peripheral is holding it.
4	STB		START byte mode bit. This bit is only applicable when the I2C module is a master. As described in version 2.1 of the Philips Semiconductors I2C-bus specification, the START byte can be used to help a slave that needs extra time to detect a START condition. When the I2C module is a slave, it ignores a START byte from a master, regardless of the value of the STB bit.
		0	The I2C module is not in the START byte mode.
		1	The I2C module is in the START byte mode. When you set the START condition bit (STT), the I2C module begins the transfer with more than just a START condition. Specifically, it generates:
			1. A START condition
			2. A START byte (0000 0001b)
			3. A dummy acknowledge clock pulse
			4. A repeated START condition
			Then, as normal, the I2C module sends the slave address that is in I2CSAR.
3	FDF		Free data format mode bit.
		0	Free data format mode is disabled. Transfers use the 7-/10-bit addressing format selected by the XA bit.
		1	Free data format mode is enabled. Transfers have the free data (no address) format described in Section 14.2.5.
			The free data format is not supported in the digital loopback mode (DLB=1).

Copyright © 2012–2019, Texas Instruments Incorporated

Table 14-5. I2C Mode Register (I2CMDR) Field Descriptions (continued)

Bit	Field	Value	Description
2-0	BC		Bit count bits. BC defines the number of bits (1 to 8) in the next data byte that is to be received or transmitted by the I2C module. The number of bits selected with BC must match the data size of the other device. Notice that when BC = 000b, a data byte has 8 bits. BC does not affect address bytes, which always have 8 bits. Note: If the bit count is less than 8, receive data is right-justified in I2CDRR(7-0), and the other bits of I2CDRR(7-0) are undefined. Also, transmit data written to I2CDXR must be right-justified.
		000	8 bits per data byte
		001	1 bit per data byte
		010	2 bits per data byte
		011	3 bits per data byte
		100	4 bits per data byte
		101	5 bits per data byte
		110	6 bits per data byte
		111	7 bits per data byte

Table 14-6. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR

RM	STT	STP	Bus Activity[(1)]	Description
0	0	0	None	No activity
0	0	1	P	STOP condition
0	1	0	S-A-D..(n)..D.	START condition, slave address, n data bytes (n = value in I2CCNT)
0	1	1	S-A-D..(n)..D-P	START condition, slave address, n data bytes, STOP condition (n = value in I2CCNT)
1	0	0	None	No activity
1	0	1	P	STOP condition
1	1	0	S-A-D-D-D.	Repeat mode transfer: START condition, slave address, continuous data transfers until STOP condition or next START condition
1	1	1	None	Reserved bit combination (No activity)

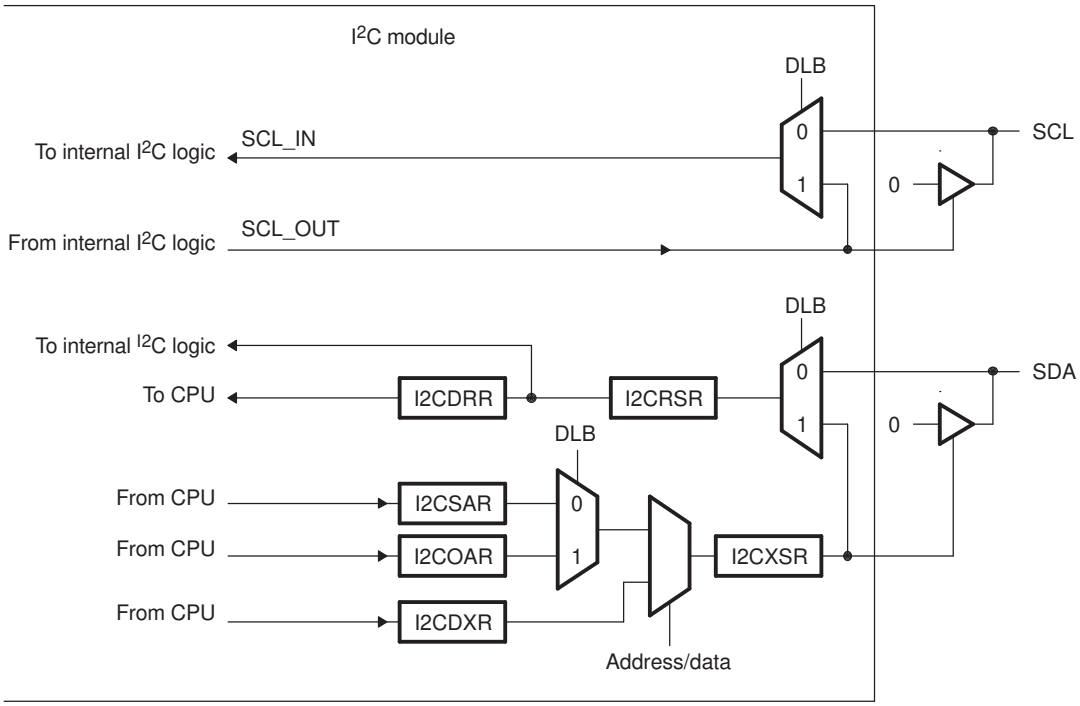
(1) S = START condition; A = Address; D = Data byte; P = STOP condition;

Table 14-7. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR

MST	FDF	I2C Module State	Function of TRX
0	0	In slave mode but not free data format mode	TRX is a don't care. Depending on the command from the master, the I2C module responds as a receiver or a transmitter.
0	1	In slave mode and free data format mode	The free data format mode requires that the I2C module remains the transmitter or the receiver throughout the transfer. TRX identifies the role of the I2C module: TRX = 1: The I2C module is a transmitter. TRX = 0: The I2C module is a receiver.
1	0	In master mode but not free data format mode	TRX = 1: The I2C module is a transmitter. TRX = 0: The I2C module is a receiver.
1	1	In master mode and free data format mode	TRX = 0: The I2C module is a receiver. TRX = 1: The I2C module is a transmitter.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 14-15. Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit



14.5.2 I2C Extended Mode Register (I2CEMDR)

The I2C extended mode register is shown in Figure 14-16 and described in Table 14-8.

Figure 14-16. I2C Extended Mode Register (I2CEMDR)

15	1	0
Reserved		BCM
R-0		R/W-1

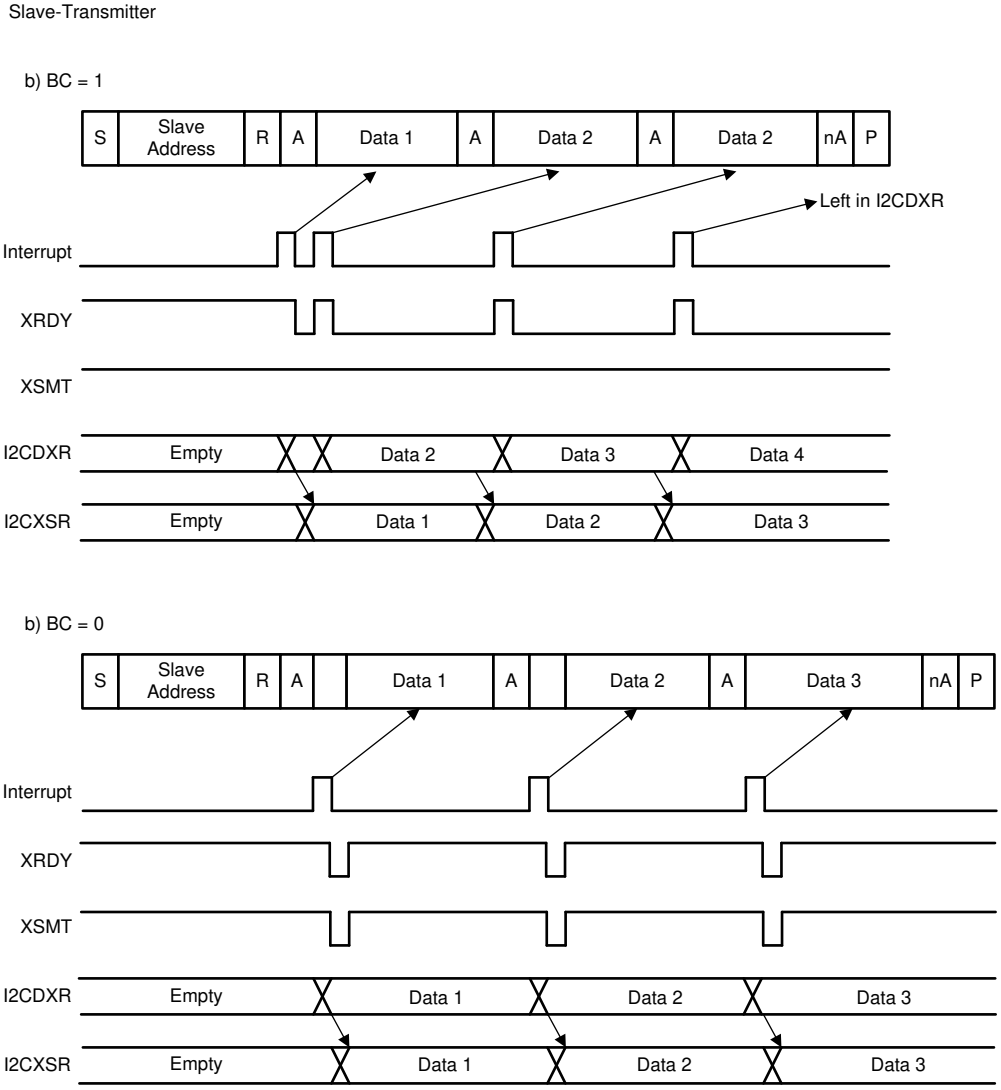
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-8. I2C Extended Mode Register (I2CEMDR) Field Descriptions

Bit	Field	Value	Description
15-1	Reserved		Any writes to these bit(s) must always have a value of 0.
0	BCM		Backwards compatibility mode. This bit affects the timing of the transmit status bits (XRDY and XSMT) in the I2CSTR register when in slave transmitter mode. See Figure 14-17 for details

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 14-17. BCM Bit, Slave Transmitter Mode



Copyright © 2012–2019, Texas Instruments Incorporated

14.5.3 I2C Interrupt Enable Register (I2CIER)

I2CIER is used by the CPU to individually enable or disable I2C interrupt requests. The bits of I2CIER are shown and described in Figure 14-18 and Table 14-9, respectively.

Figure 14-18. I2C Interrupt Enable Register (I2CIER)

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved	AAS	SCD	XRDY	RRDY	ARDY	NACK	AL
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 14-9. I2C Interrupt Enable Register (I2CIER) Field Descriptions

Bit	Field	Value	Description
15-7	Reserved		These reserved bit locations are always read as zeros. A value written to this field has no effect.
6	AAS		Addressed as slave interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled
5	SCD		Stop condition detected interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled
4	XRDY		Transmit-data-ready interrupt enable bit. This bit should not be set when using FIFO mode.
		0	Interrupt request disabled
		1	Interrupt request enabled
3	RRDY		Receive-data-ready interrupt enable bit. This bit should not be set when using FIFO mode.
		0	Interrupt request disabled
		1	Interrupt request enabled
2	ARDY		Register-access-ready interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled
1	NACK		No-acknowledgment interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled
0	AL		Arbitration-lost interrupt enable bit
		0	Interrupt request disabled
		1	Interrupt request enabled

14.5.4 I2C Status Register (I2CSTR)

The I2C status register (I2CSTR) is a 16-bit register used to determine which interrupt has occurred and to read status information. The bits of I2CSTR are shown and described in Figure 14-19 and Table 14-10, respectively.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 14-19. I2C Status Register (I2CSTR)

15	14	13	12	11	10	9	8
Reserved	SDIR	NACKSNT	BB	RSFULL	XSMT	AAS	AD0
R-0	R/W1C-0	R/W1C-0	R-0	R-0	R-1	R-0	R-0
7	6	5	4	3	2	1	0
Reserved		SCD	XRDY	RRDY	ARDY	NACK	AL
R-0		R/W1C-0	R-1	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; W1C = Write 1 to clear (writing 0 has no effect); R = Read only; -*n* = value after reset

Table 14-10. I2C Status Register (I2CSTR) Field Descriptions

Bit	Field	Value	Description
15	Reserved	0	This reserved bit location is always read as zeros. A value written to this bit has no effect.
14	SDIR		Slave direction bit
		0	I2C is not addressed as a slave transmitter. SDIR is cleared by one of the following events: • It is manually cleared. To clear this bit, write a 1 to it. • Digital loopback mode is enabled. • A START or STOP condition occurs on the I2C bus.
		1	I2C is addressed as a slave transmitter.
13	NACKSNT		NACK sent bit. This bit is used when the I2C module is in the receiver mode. One instance in which NACKSNT is affected is when the NACK mode is used (see the description for NACKMOD in Section 14.5.1).
		0	NACK not sent. NACKSNT bit is cleared by any one of the following events: • It is manually cleared. To clear this bit, write a 1 to it. • The I2C module is reset (either when 0 is written to the IRS bit of I2CMDR or when the whole device is reset).
		1	NACK sent: A no-acknowledge bit was sent during the acknowledge cycle on the I2C-bus.
12	BB		Bus busy bit. BB indicates whether the I2C-bus is busy or is free for another data transfer. See the paragraph following the table for more information.
		0	Bus free. BB is cleared by any one of the following events: • The I2C module receives or transmits a STOP bit (bus free). • The I2C module is reset.
		1	Bus busy: The I2C module has received or transmitted a START bit on the bus.
11	RSFULL		Receive shift register full bit. RSFULL indicates an overrun condition during reception. Overrun occurs when new data is received into the shift register (I2CRSR) and the old data has not been read from the receive register (I2CDRR). As new bits arrive from the SDA pin, they overwrite the bits in I2CRSR. The new data will not be copied to ICDRR until the previous data is read.
		0	No overrun detected. RSFULL is cleared by any one of the following events: • I2CDRR is read is read by the CPU. Emulator reads of the I2CDRR do not affect this bit. • The I2C module is reset.
		1	Overrun detected
10	XSMT		Transmit shift register empty bit. XSMT = 0 indicates that the transmitter has experienced underflow. Underflow occurs when the transmit shift register (I2CXSR) is empty but the data transmit register (I2CDXR) has not been loaded since the last I2CDXR-to-I2CXSR transfer. The next I2CDXR-to-I2CXSR transfer will not occur until new data is in I2CDXR. If new data is not transferred in time, the previous data may be re-transmitted on the SDA pin.
		0	Underflow detected (empty)
		1	No underflow detected (not empty). XSMT is set by one of the following events: • Data is written to I2CDXR. • The I2C module is reset
9	AAS		Addressed-as-slave bit
		0	In the 7-bit addressing mode, the AAS bit is cleared when receiving a NACK, a STOP condition, or a repeated START condition. In the 10-bit addressing mode, the AAS bit is cleared when receiving a NACK, a STOP condition, or by a slave address different from the I2C peripheral's own slave address.
		1	The I2C module has recognized its own slave address or an address of all zeros (general call).

Copyright © 2012–2019, Texas Instruments Incorporated

Table 14-10. I2C Status Register (I2CSTR) Field Descriptions (continued)

Bit	Field	Value	Description
8	AD0		Address 0 bits
		0	AD0 has been cleared by a START or STOP condition.
		1	An address of all zeros (general call) is detected.
7-6	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
5	SCD		Stop condition detected bit. SCD is set when the I2C sends or receives a STOP condition. The I2C module delays clearing of the I2CMDR[STP] bit until the SCD bit is set.
		0	STOP condition not detected since SCD was last cleared. SCD is cleared by any one of the following events: • I2CISRC is read by the CPU when it contains the value 110b (stop condition detected). Emulator reads of the I2CISRC do not affect this bit. • SCD is manually cleared. To clear this bit, write a 1 to it. • The I2C module is reset.
		1	A STOP condition has been detected on the I2C bus.
4	XRDY		Transmit-data-ready interrupt flag bit. When not in FIFO mode, XRDY indicates that the data transmit register (I2CDXR) is ready to accept new data because the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR). The CPU can poll XRDY or use the XRDY interrupt request (see Section 14.3.1). When in FIFO mode, use TXFFINT instead.
		0	I2CDXR not ready. XRDY is cleared when data is written to I2CDXR.
		1	I2CDXR ready: Data has been copied from I2CDXR to I2CXSR. XRDY is also forced to 1 when the I2C module is reset.
3	RRDY		Receive-data-ready interrupt flag bit. When not in FIFO mode, RRDY indicates that the data receive register (I2CDRR) is ready to be read because data has been copied from the receive shift register (I2CRSR) to I2CDRR. The CPU can poll RRDY or use the RRDY interrupt request (see Section 14.3.1). When in FIFO mode, use RXFFINT instead.
		0	I2CDRR not ready. RRDY is cleared by any one of the following events: • I2CDRR is read by the CPU. Emulator reads of the I2CDRR do not affect this bit. • RRDY is manually cleared. To clear this bit, write a 1 to it. • The I2C module is reset.
		1	I2CDRR ready: Data has been copied from I2CRSR to I2CDRR.
2	ARDY		Register-access-ready interrupt flag bit (only applicable when the I2C module is in the master mode). ARDY indicates that the I2C module registers are ready to be accessed because the previously programmed address, data, and command values have been used. The CPU can poll ARDY or use the ARDY interrupt request (see Section 14.3.1)
		0	The registers are not ready to be accessed. ARDY is cleared by any one of the following events: • The I2C module starts using the current register contents. • ARDY is manually cleared. To clear this bit, write a 1 to it. • The I2C module is reset.
		1	The registers are ready to be accessed. In the nonrepeat mode (RM = 0 in I2CMDR): If STP = 0 in I2CMDR, the ARDY bit is set when the internal data counter counts down to 0. If STP = 1, ARDY is not affected (instead, the I2C module generates a STOP condition when the counter reaches 0). In the repeat mode (RM = 1): ARDY is set at the end of each byte transmitted from I2CDXR.
1	NACK		No-acknowledgment interrupt flag bit. NACK applies when the I2C module is a transmitter (master or slave). NACK indicates whether the I2C module has detected an acknowledge bit (ACK) or a no-acknowledge bit (NACK) from the receiver. The CPU can poll NACK or use the NACK interrupt request (see Section 14.3.1).
		0	ACK received/NACK not received. This bit is cleared by any one of the following events: • An acknowledge bit (ACK) has been sent by the receiver. • NACK is manually cleared. To clear this bit, write a 1 to it. • The CPU reads the interrupt source register (I2CISRC) and the register contains the code for a NACK interrupt. Emulator reads of the I2CISRC do not affect this bit. • The I2C module is reset.
		1	NACK bit received. The hardware detects that a no-acknowledge (NACK) bit has been received. Note: While the I2C module performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgment.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 14-10. I2C Status Register (I2CSTR) Field Descriptions (continued)

Bit	Field	Value	Description
0	AL		Arbitration-lost interrupt flag bit (only applicable when the I2C module is a master-transmitter). AL primarily indicates when the I2C module has lost an arbitration contest with another master-transmitter. The CPU can poll AL or use the AL interrupt request (see Section 14.3.1)
		0	Arbitration not lost. AL is cleared by any one of the following events: • AL is manually cleared. To clear this bit, write a 1 to it. • The CPU reads the interrupt source register (I2CISRC) and the register contains the code for an AL interrupt. Emulator reads of the I2CISRC do not affect this bit. • The I2C module is reset.
		1	Arbitration lost. AL is set by any one of the following events: • The I2C module senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously. • The I2C module attempts to start a transfer while the BB (bus busy) bit is set to 1. When AL becomes 1, the MST and STP bits of I2CMDR are cleared, and the I2C module becomes a slave-receiver.

The I2C peripheral cannot detect a START or STOP condition when it is in reset, i.e. the IRS bit is set to 0. Therefore, the BB bit will remain in the state it was at when the peripheral was placed in reset. The BB bit will stay in that state until the I2C peripheral is taken out of reset, i.e. the IRS bit is set to 1, and a START or STOP condition is detected on the I2C bus.

Follow these steps before initiating the first data transfer with I2C :

1. After taking the I2C peripheral out of reset by setting the IRS bit to 1, wait a certain period to detect the actual bus status before starting the first data transfer. Set this period larger than the total time taken for the longest data transfer in the application. By waiting for a period of time after I2C comes out of reset, users can ensure that at least one START or STOP condition will have occurred on the I2C bus, and been captured by the BB bit. After this period, the BB bit will correctly reflect the state of the I2C bus.
2. Check the BB bit and verify that BB=0 (bus not busy) before proceeding.
3. Begin data transfers.

Not resetting the I2C peripheral in between transfers ensures that the BB bit reflects the actual bus status. If users must reset the I2C peripheral in between transfers, repeat steps 1 through 3 every time the I2C peripheral is taken out of reset.

14.5.5 I2C Interrupt Source Register (I2CISRC)

The I2C interrupt source register (I2CISRC) is a 16-bit register used by the CPU to determine which event generated the I2C interrupt. For more information about these events, see the descriptions of the I2C interrupt requests in Table 14-3.

Figure 14-20. I2C Interrupt Source Register (I2CISRC)

15 12	11 8	7 3	2 0
Reserved	Reserved	Reserved	INTCODE
R-0	R/W-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-11. I2C Interrupt Source Register (I2CISRC) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved		These reserved bit locations are always read as zeros. A value written to this field has no effect.
11-8	Reserved		These reserved bit locations should always be written as zeros.
7-3	Reserved		These reserved bit locations are always read as zeros. A value written to this field has no effect.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 14-11. I2C Interrupt Source Register (I2CISRC) Field Descriptions (continued)

Bit	Field	Value	Description
2-0	INTCODE		Interrupt code bits. The binary code in INTCODE indicates the event that generated an I2C interrupt.
		000	None
		001	Arbitration lost
		010	No-acknowledgment condition detected
		011	Registers ready to be accessed
		100	Receive data ready
		101	Transmit data ready
		110	Stop condition detected
		111	Addressed as slave
			A CPU read will clear this field. If another lower priority interrupt is pending and enabled, the value corresponding to that interrupt will then be loaded. Otherwise, the value will stay cleared.
			In the case of an arbitration lost, a no-acknowledgment condition detected, or a stop condition detected, a CPU read will also clear the associated interrupt flag bit in the I2CSTR register.
			Emulator reads will not affect the state of this field or of the status bits in the I2CSTR register.

14.5.6 I2C Prescaler Register (I2CPSC)

The I2C prescaler register (I2CPSC) is a 16-bit register (see Figure 14-21) used for dividing down the I2C input clock to obtain the desired module clock for the operation of the I2C module. See the device-specific data manual for the supported range of values for the module clock frequency. Table 14-12 lists the bit descriptions. For more details about the module clock, see Section 14.1.3.

IPSC must be initialized while the I2C module is in reset (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

Figure 14-21. I2C Prescaler Register (I2CPSC)

15	8	7	0
Reserved		IPSC	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-12. I2C Prescaler Register (I2CPSC) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved		These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	IPSC		I2C prescaler divide-down value. IPSC determines how much the CPU clock is divided to create the module clock of the I2C module: module clock frequency = I2C input clock frequency/(IPSC + 1) Note: IPSC must be initialized while the I2C module is in reset (IRS = 0 in I2CMDR).

NOTE: To meet all of the I2C protocol timing specifications, the module clock must be configured between 7-12 MHz.

14.5.7 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)

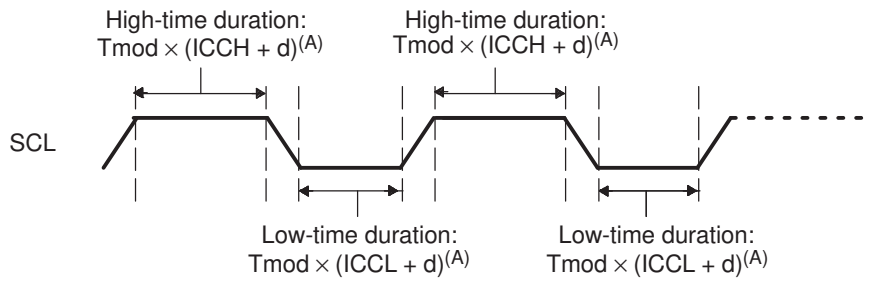
As explained in Section 14.1.3, when the I2C module is a master, the module clock is divided down for use as the master clock on the SCL pin. As shown in Figure 14-22, the shape of the master clock depends on two divide-down values:

- ICCL in I2CCLKL (summarized by Figure 14-23 and Table 14-13). For each master clock cycle, ICCL determines the amount of time the signal is low.
- ICCH in I2CCLKH (summarized by Figure 14-24 and Table 14-14). For each master clock cycle, ICCH

Copyright © 2012–2019, Texas Instruments Incorporated

determines the amount of time the signal is high.

Figure 14-22. The Roles of the Clock Divide-Down Values (ICCL and ICCH)



A As described in Section 14.5.7.1, Tmod is the module clock period, and d is 5, 6, or 7.

Figure 14-23. I2C Clock Low-Time Divider Register (I2CCLKL)

15	0
ICCL	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-13. I2C Clock Low-Time Divider Register (I2CCLKL) Field Description

Bit	Field	Value	Description
15-0	ICCL		Clock low-time divide-down value. To produce the low-time duration of the master clock, the period of the module clock is multiplied by (ICCL + d). d is 5, 6, or 7 as described in Section 14.5.7.1. Note: These bits must be set to a non-zero value for proper I2C clock operation.

Figure 14-24. I2C Clock High-Time Divider Register (I2CCLKH)

15	0
ICCH	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-14. I2C Clock High-Time Divider Register (I2CCLKH) Field Description

Bit	Field	Value	Description
15-0	ICCH		Clock high-time divide-down value. To produce the high-time duration of the master clock, the period of the module clock is multiplied by (ICCH + d). d is 5, 6, or 7 as described in Section 14.5.7.1. Note: These bits must be set to a non-zero value for proper I2C clock operation.

14.5.7.1 Formula for the Master Clock Period

The period of the master clock (Tmst) is a multiple of the period of the module clock (Tmod):

$$T_{mst} = T_{mod} \times [(ICCL + d) + (ICCH + d)]$$

$$T_{mst} = \frac{(IPSC + 1)\ [(ICCL + d) + (ICCH + d)]}{I2C\ \ input\ \ clock\ \ frequency}$$

where d depends on the divide-down value IPSC, as shown in Table 14-15. IPSC is described in Section 14.5.6.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 14-15. Dependency of Delay d on the Divide-Down Value IPSC

IPSC	d
0	7
1	6
Greater than 1	5

14.5.8 I2C Slave Address Register (I2CSAR)

The I2C slave address register (I2CSAR) is a register for storing the next slave address that will be transmitted by the I2C module when it is a master. It is a 16-bit register with the format shown in Figure 14-25. As described in Table 14-16, the SAR field of I2CSAR contains a 7-bit or 10-bit slave address. When the I2C module is not using the free data format (FDF = 0 in I2CMDR), it uses this address to initiate data transfers with a slave or slaves. When the address is nonzero, the address is for a particular slave. When the address is 0, the address is a general call to all slaves. If the 7-bit addressing mode is selected (XA = 0 in I2CMDR), only bits 6-0 of I2CSAR are used; write 0s to bits 9-7.

Figure 14-25. I2C Slave Address Register (I2CSAR)

15	10	9	0
Reserved		SAR	
R-0		R/W-3FFh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-16. I2C Slave Address Register (I2CSAR) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved		These reserved bit locations are always read as zeros. A value written to this field has no effect.
9-0	SAR		In 7-bit addressing mode (XA = 0 in I2CMDR):
		00h-7Fh	Bits 6-0 provide the 7-bit slave address that the I2C module transmits when it is in the master-transmitter mode. Write 0s to bits 9-7.
			In 10-bit addressing mode (XA = 1 in I2CMDR):
		000h-3FFh	Bits 9-0 provide the 10-bit slave address that the I2C module transmits when it is in the master-transmitter mode.

14.5.9 I2C Own Address Register (I2COAR)

The I2C own address register (I2COAR) is a 16-bit register. Figure 14-26 shows the format of I2COAR, and Table 14-17 describes its bit fields. The I2C module uses this register to specify its own slave address, which distinguishes it from other slaves connected to the I2C-bus. If the 7-bit addressing mode is selected (XA = 0 in I2CMDR), only bits 6-0 are used; write 0s to bits 9-7.

Figure 14-26. I2C Own Address Register (I2COAR)

15	10	9	0
Reserved		OAR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-17. I2C Own Address Register (I2COAR) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved		These reserved bit locations are always read as zeros. A value written to this field has no effect.
9-0	OAR		In 7-bit addressing mode (XA = 0 in I2CMDR):
		00h-7Fh	Bits 6-0 provide the 7-bit slave address of the I2C module. Write 0s to bits 9-7.
			In 10-bit addressing mode (XA = 1 in I2CMDR):
		000h-3FFh	Bits 9-0 provide the 10-bit slave address of the I2C module.

Copyright © 2012–2019, Texas Instruments Incorporated

14.5.10 I2C Data Count Register (I2CCNT)

I2CCNT is a 16-bit register used to indicate how many data bytes to transfer when the I2C module is configured as a transmitter, or to receive when configured as a master receiver. In the repeat mode (RM = 1), I2CCNT is not used. The bits of I2CCNT are shown and described in Figure 14-27 and Table 14-18, respectively.

The value written to I2CCNT is copied to an internal data counter. The internal data counter is decremented by 1 for each byte transferred (I2CCNT remains unchanged). If a STOP condition is requested in the master mode (STP = 1 in I2CMDR), the I2C module terminates the transfer with a STOP condition when the countdown is complete (that is, when the last byte has been transferred).

Figure 14-27. I2C Data Count Register (I2CCNT)

15	0
ICDC	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-18. I2C Data Count Register (I2CCNT) Field Descriptions

Bit	Field	Value	Description
15-0	ICDC		Data count value. ICDC indicates the number of data bytes to transfer or receive. The value in I2CCNT is a don't care when the RM bit in I2CMDR is set to 1.
		0000h	The start value loaded to the internal data counter is 65536.
		0001h-FFFFh	The start value loaded to internal data counter is 1-65535.

14.5.11 I2C Data Receive Register (I2CDRR)

I2CDRR (see Figure 14-28 and Table 14-19) is a 16-bit register used by the CPU to read received data. The I2C module can receive a data byte with 1 to 8 bits. The number of bits is selected with the bit count (BC) bits in I2CMDR. One bit at a time is shifted in from the SDA pin to the receive shift register (I2CRSR). When a complete data byte has been received, the I2C module copies the data byte from I2CRSR to I2CDRR. The CPU cannot access I2CRSR directly.

If a data byte with fewer than 8 bits is in I2CDRR, the data value is right-justified, and the other bits of I2CDRR(7-0) are undefined. For example, if BC = 011 (3-bit data size), the receive data is in I2CDRR(2-0), and the content of I2CDRR(7-3) is undefined.

When in the receive FIFO mode, the I2CDRR register acts as the receive FIFO buffer.

Figure 14-28. I2C Data Receive Register (I2CDRR)

15	8	7	0
Reserved		DATA	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-19. I2C Data Receive Register (I2CDRR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved		These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	DATA		Receive data

14.5.12 I2C Data Transmit Register (I2CDXR)

The CPU writes transmit data to I2CDXR (see Figure 14-29 and Table 14-20). This 16-bit register accepts a data byte with 1 to 8 bits. Before writing to I2CDXR, specify how many bits are in a data byte by loading the appropriate value into the bit count (BC) bits of I2CMDR. When writing a data byte with fewer than 8 bits, make sure the value is right-aligned in I2CDXR.

Copyright © 2012–2019, Texas Instruments Incorporated

After a data byte is written to I2CDXR, the I2C module copies the data byte to the transmit shift register (I2CXSR). The CPU cannot access I2CXSR directly. From I2CXSR, the I2C module shifts the data byte out on the SDA pin, one bit at a time.

When in the transmit FIFO mode, the I2CDXR register acts as the transmit FIFO buffer.

Figure 14-29. I2C Data Transmit Register (I2CDXR)

15	8	7	0
Reserved		DATA	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-20. I2C Data Transmit Register (I2CDXR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved		These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	DATA		Transmit data

14.5.13 I2C Transmit FIFO Register (I2CFFTX)

The I2C transmit FIFO register (I2CFFTX) is a 16-bit register that contains the I2C FIFO mode enable bit as well as the control and status bits for the transmit FIFO mode of operation on the I2C peripheral. The bit fields are shown in Figure 14-30 and described in Table 14-21.

Figure 14-30. I2C Transmit FIFO Register (I2CFFTX)

15	14	13	12	11	10	9	8
Reserved	I2CFFEN	TXFFRST	TXFFST4	TXFFST3	TXFFST2	TXFFST1	TXFFST0
R-0	R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
TXFFINT	TXFFINTCLR	TXFFIENA	TXFFIL4	TXFFIL3	TXFFIL2	TXFFIL1	TXFFIL0
R-0	R/W1C-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-21. I2C Transmit FIFO Register (I2CFFTX) Field Descriptions

Bit	Field	Value	Description
15	Reserved		Reserved. Reads will return a 0, writes have no effect.
14	I2CFFEN		I2C FIFO mode enable bit. This bit must be enabled for either the transmit or the receive FIFO to operate correctly.
		0	Disable the I2C FIFO mode.
		1	Enable the I2C FIFO mode.
13	TXFFRST		I2C transmit FIFO reset bit.
		0	Reset the transmit FIFO pointer to 0000 and hold the transmit FIFO in the reset state.
		1	Enable the transmit FIFO operation.
12-8	TXFFST4-0		Contains the status of the transmit FIFO:
		10000	Transmit FIFO contains 16 bytes.
		0xxxx	Transmit FIFO contains xxxx bytes.
		00000	Transmit FIFO is empty.
			Note: Since these bits are reset to zero, the transmit FIFO interrupt flag will be set when the transmit FIFO operation is enabled and the I2C is taken out of reset. This will generate a transmit FIFO interrupt if enabled. To avoid any detrimental effects from this, write a one to the TXFFINTCLR once the transmit FIFO operation is enabled and the I2C is taken out of reset.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 14-21. I2C Transmit FIFO Register (I2CFFTX) Field Descriptions (continued)

Bit	Field	Value	Description
7	TXFFINT		Transmit FIFO interrupt flag. This bit cleared by a CPU write of a 1 to the TXFFINTCLR bit. If the TXFFIENA bit is set, this bit will generate an interrupt when it is set.
		0	Transmit FIFO interrupt condition has not occurred.
		1	Transmit FIFO interrupt condition has occurred.
6	TXFFINTCLR		Transmit FIFO interrupt flag clear bit.
		0	Writes of zeros have no effect. Reads return a 0.
		1	Writing a 1 to this bit clears the TXFFINT flag.
5	TXFFIENA		Transmit FIFO interrupt enable bit.
		0	Disabled. TXFFINT flag does not generate an interrupt when set.
		1	Enabled. TXFFINT flag does generate an interrupt when set.
4-0	TXFFIL4-0		Transmit FIFO interrupt level.
			These bits set the status level that will set the transmit interrupt flag. When the TXFFST4-0 bits reach a value equal to or less than these bits, the TXFFINT flag will be set. This will generate an interrupt if the TXFFIENA bit is set.

14.5.14 I2C Receive FIFO Register (I2CFFRX)

The I2C receive FIFO register (I2CFFRX) is a 16-bit register that contains the control and status bits for the receive FIFO mode of operation on the I2C peripheral. The bit fields are shown in Figure 14-31 and described in Table 14-22.

Figure 14-31. I2C Receive FIFO Register (I2CFFRX)

15	14	13	12	11	10	9	8
Reserved		RXFFRST	RXFFST4	RXFFST3	RXFFST2	RXFFST1	RXFFST0
R-0		R/W-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RXFFINT	RXFFINTCLR	RXFFIENA	RXFFIL4	RXFFIL3	RXFFIL2	RXFFIL1	RXFFIL0
R-0	R/W1C-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 14-22. I2C Receive FIFO Register (I2CFFRX) Field Descriptions

Bit	Field	Value	Description
15-14	Reserved		Reserved. Reads will return a 0, writes have no effect.
13	RXFFRST		I2C receive FIFO reset bit
		0	Reset the receive FIFO pointer to 0000 and hold the receive FIFO in the reset state.
		1	Enable the receive FIFO operation.
12-8	RXFFST4-0		Contains the status of the receive FIFO:
		10000	Receive FIFO contains 16 bytes
		0xxxx	Receive FIFO contains xxxx bytes
		00000	Receive FIFO is empty.
7	RXFFINT		Receive FIFO interrupt flag. This bit cleared by a CPU write of a 1 to the RXFFINTCLR bit. If the RXFFIENA bit is set, this bit will generate an interrupt when it is set.
		0	Receive FIFO interrupt condition has not occurred.
		1	Receive FIFO interrupt condition has occurred.
6	RXFFINTCLR		Receive FIFO interrupt flag clear bit.
		0	Writes of zeros have no effect. Reads return a zero.
		1	Writing a 1 to this bit clears the RXFFINT flag.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 14-22. I2C Receive FIFO Register (I2CFFRX) Field Descriptions (continued)

Bit	Field	Value	Description
5	RXFFIENA		Receive FIFO interrupt enable bit.
		0	Disabled. RXFFINT flag does not generate an interrupt when set.
		1	Enabled. RXFFINT flag does generate an interrupt when set.
4-0	RXFFIL4-0		Receive FIFO interrupt level. These bits set the status level that will set the receive interrupt flag. When the RXFFST4-0 bits reach a value equal to or greater than these bits, the RXFFINT flag is set. This will generate an interrupt if the RXFFIENA bit is set. Note: Since these bits are reset to zero, the receive FIFO interrupt flag will be set if the receive FIFO operation is enabled and the I2C is taken out of reset. This will generate a receive FIFO interrupt if enabled. To avoid this, modify these bits on the same instruction as or prior to setting the RXFFRST bit.

Copyright © 2012–2019, Texas Instruments Incorporated

C28 Multichannel Buffered Serial Port (McBSP)

This chapter describes the multichannel buffered serial port (McBSP). This device provides one high-speed multichannel buffered serial port (McBSP) that allows direct interface to codecs and other devices in a system. .

Topic **Page**

15.1 Overview 1037
15.2 Clocking and Framing Data 1042
15.3 Frame Phases 1044
15.4 McBSP Sample Rate Generator 1049
15.5 McBSP Exception/Error Conditions 1056
15.6 Multichannel Selection Modes 1064
15.7 SPI Operation Using the Clock Stop Mode 1071
15.8 Receiver Configuration 1078
15.9 Transmitter Configuration 1097
15.10 Emulation and Reset Considerations 1114
15.11 Data Packing Examples 1116
15.12 McBSP Registers 1118

Copyright © 2012–2019, Texas Instruments Incorporated

15.1 Overview

The McBSP consists of a data-flow path and a control path connected to external devices by six pins as shown in Figure 15-1.

Data is communicated to devices interfaced with the McBSP via the data transmit (DX) pin for transmission and via the data receive (DR) pin for reception. Control information in the form of clocking and frame synchronization is communicated via the following pins: CLKX (transmit clock), CLKR (receive clock), FSX (transmit frame synchronization), and FSR (receive frame synchronization).

The CPU and the DMA controller communicate with the McBSP through 16-bit-wide registers accessible via the internal peripheral bus. The CPU or the DMA controller writes the data to be transmitted to the data transmit registers (DXR1, DXR2). Data written to the DXRs is shifted out to DX via the transmit shift registers (XSR1, XSR2). Similarly, receive data on the DR pin is shifted into the receive shift registers (RSR1, RSR2) and copied into the receive buffer registers (RBR1, RBR2). The contents of the RBRs is then copied to the DRRs, which can be read by the CPU or the DMA controller. This allows simultaneous movement of internal and external data communications.

DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted) if the serial word length is 8 bits, 12 bits, or 16 bits. For larger word lengths, these registers are needed to hold the most significant bits.

The frame and clock loopback is implemented at chip level to enable CLKX and FSX to drive CLKR and FSR. If the loop-back is enabled, the CLKR and FSR get their signals from the CLKX and FSX pads; instead of the CLKR and FSR pins.

15.1.1 Features of the McBSP

The McBSP features:

- Full-duplex communication
- Double-buffered transmission and triple-buffered reception, allowing a continuous data stream
- Independent clocking and framing for reception and transmission
- The capability to send interrupts to the CPU and to send DMA events to the DMA controller
- 128 channels for transmission and reception
- Multichannel selection modes that enable or disable block transfers in each of the channels
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- Support for external generation of clock signals and frame-synchronization signals
- A programmable sample rate generator for internal generation and control of clock signals and frame-synchronization signals
- Programmable polarity for frame-synchronization pulses and clock signals
- Direct interface to:
  - T1/E1 framers
  - IOM-2 compliant devices
  - AC97-compliant devices (the necessary multiphase frame capability is provided)
  - I2S compliant devices
  - SPI devices
- A wide selection of data sizes: 8, 12, 16, 20, 24, and 32 bits

NOTE: A value of the chosen data size is referred to as a *serial word* or *word* throughout the McBSP documentation. Elsewhere, *word* is used to describe a 16-bit value.

- μ-law and A-law companding
- The option of transmitting/receiving 8-bit data with the LSB first
- Status bits for flagging exception/error conditions
- ABIS mode is not supported.

Copyright © 2012–2019, Texas Instruments Incorporated

15.1.2 McBSP Pins/Signals

Table 15-1 describes the McBSP interface pins and some internal signals.

Table 15-1. McBSP Interface Pins/Signals

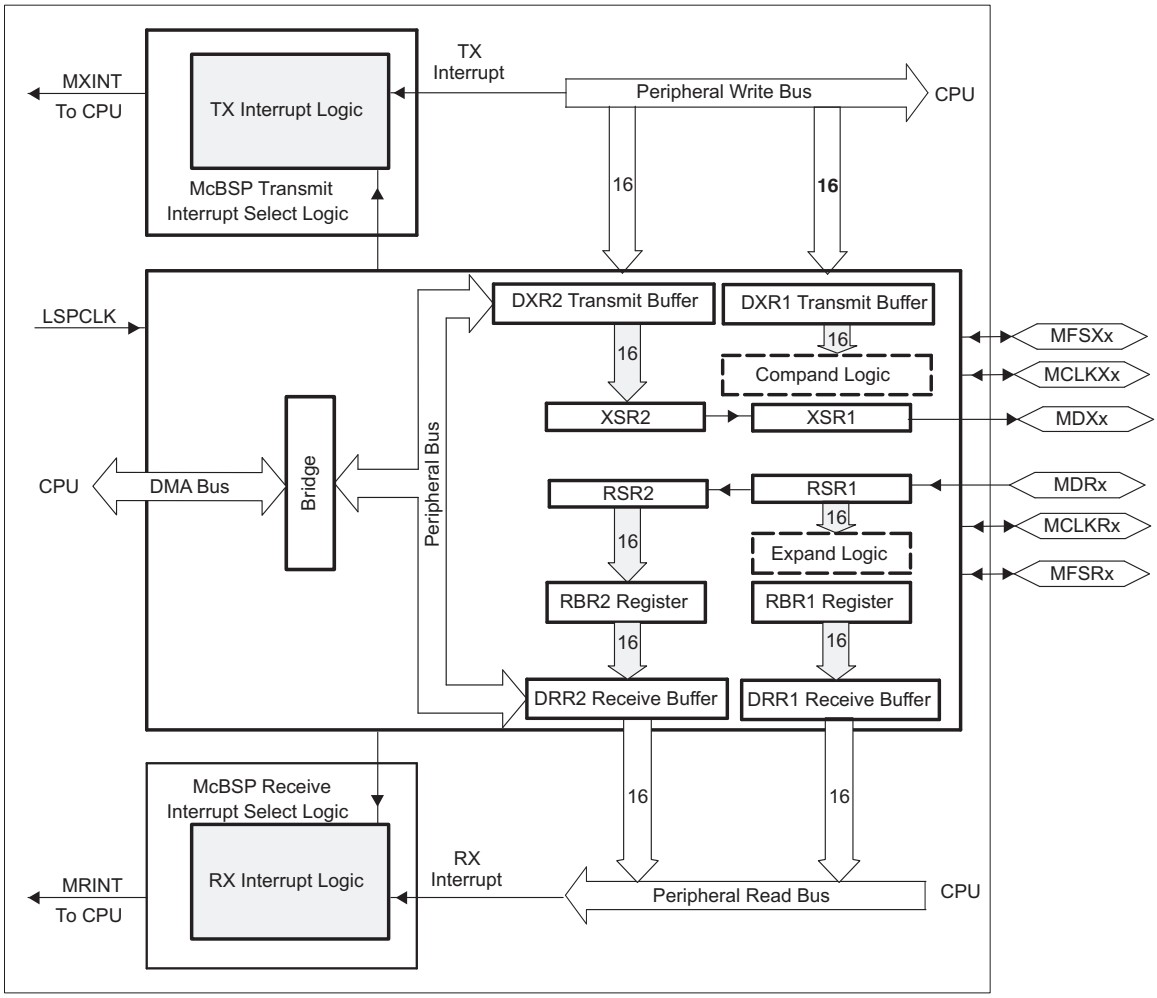
McBSP-A Pin	Type	Description
MCLKRA	I/O	Supplying or reflecting the receive clock; supplying the input clock of the sample rate generator
MCLKXA	I/O	Supplying or reflecting the transmit clock; supplying the input clock of the sample rate generator
MDRA	I	Serial data receive pin
MDXA	O	Serial data transmit pin
MFSRA	I/O	Supplying or reflecting the receive frame-sync signal; controlling sample rate generator synchronization for the case when GSYNC = 1 (see Section 15.4.3)
MFSXA	I/O	Supplying or reflecting the transmit frame-sync signal
CPU Interrupt Signals		
MRINT		Receive interrupt to CPU
MXINT		Transmit interrupt to CPU
DMA Events		
REVT		Receive synchronization event to DMA
XEVT		Transmit synchronization event to DMA

Copyright © 2012–2019, Texas Instruments Incorporated

15.1.2.1 McBSP Generic Block Diagram

The McBSP consists of a data-flow path and a control path connected to external devices by six pins as shown in Figure 15-1. The figure and the text in this section use generic pin names.

Figure 15-1. Conceptual Block Diagram of the McBSP



A Not available in all devices. See the device-specific data sheet

15.1.3 McBSP Operation

This section addresses the following topics:

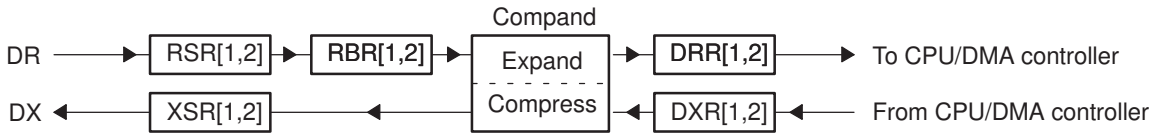
- Data transfer process
- Companding (compressing and expanding) data
- Clocking and framing data
- Frame phases
- McBSP reception
- McBSP transmission
- Interrupts and DMA events generated by McBSPs

Copyright © 2012–2019, Texas Instruments Incorporated

15.1.4 Data Transfer Process of McBSP

Figure 15-2 shows a diagram of the McBSP data transfer paths. The McBSP receive operation is triple-buffered, and transmit operation is double-buffered. The use of registers varies, depending on whether the defined length of each serial word is 16 bits.

Figure 15-2. McBSP Data Transfer Paths



15.1.4.1 Data Transfer Process for Word Length of 8, 12, or 16 Bits

If the word length is 16 bits or smaller, only one 16-bit register is needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted).

Receive data arrives on the DR pin and is shifted into receive shift register 1 (RSR1). Once a full word is received, the content of RSR1 is copied to receive buffer register 1 (RBR1) if RBR1 is not full with previous data. RBR1 is then copied to data receive register 1 (DRR1), unless the previous content of DRR1 has not been read by the CPU or the DMA controller. If the companding feature of the McBSP is implemented, the required word length is 8 bits and receive data is expanded into the appropriate format before being passed from RBR1 to DRR1. For more details about reception, see Section 15.3.5.

Transmit data is written by the CPU or the DMA controller to data transmit register 1 (DXR1). If there is no previous data in transmit shift register (XSR1), the value in DXR1 is copied to XSR1; otherwise, DXR1 is copied to XSR1 when the last bit of the previous data is shifted out on the DX pin. If selected, the companding module compresses 16-bit data into the appropriate 8-bit format before passing it to XSR1. After transmit frame synchronization, the transmitter begins shifting bits from XSR1 to the DX pin. For more details about transmission, see Section 15.3.6.

15.1.4.2 Data Transfer Process for Word Length of 20, 24, or 32 Bits

If the word length is larger than 16 bits, two 16-bit registers are needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are needed to hold the most significant bits.

Receive data arrives on the DR pin and is shifted first into RSR2 and then into RSR1. Once the full word is received, the contents of RSR2 and RSR1 are copied to RBR2 and RBR1, respectively, if RBR1 is not full. Then the contents of RBR2 and RBR1 are copied to DRR2 and DRR1, respectively, unless the previous content of DRR1 has not been read by the CPU or the DMA controller. The CPU or the DMA controller must read data from DRR2 first and then from DRR1. When DRR1 is read, the next RBR-to-DRR copy occurs. For more details about reception, see Section 15.3.5.

For transmission, the CPU or the DMA controller must write data to DXR2 first and then to DXR1. When new data arrives in DXR1, if there is no previous data in XSR1, the contents of DXR2 and DXR1 are copied to XSR2 and XSR1, respectively; otherwise, the contents of the DXRs are copied to the XSRs when the last bit of the previous data is shifted out on the DX pin. After transmit frame synchronization, the transmitter begins shifting bits from the XSRs to the DX pin. For more details about transmission, see Section 15.3.6.

15.1.5 Companding (Compressing and Expanding) Data

Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

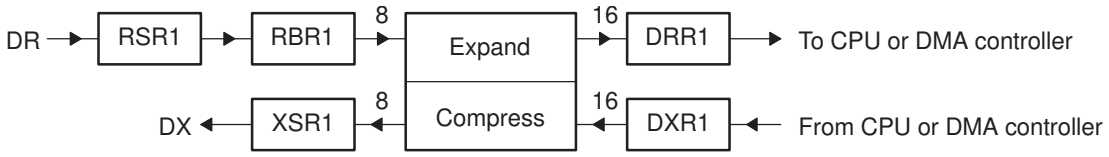
A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

Copyright © 2012–2019, Texas Instruments Incorporated

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 15-3 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to twos-complement format.

Figure 15-3. Companding Processes

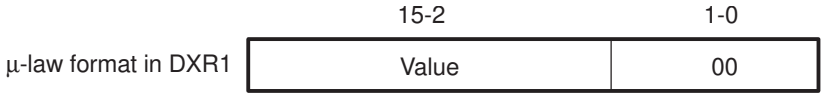


15.1.5.1 Companding Formats

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The receive sign-extension and justification mode specified in RJUST is ignored when companding is used.

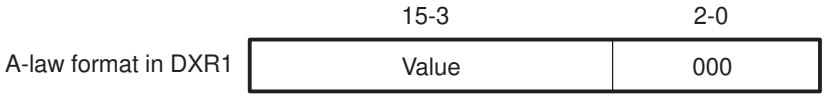
For transmission using μ-law compression, the 14 data bits must be left-justified in DXR1 and that the remaining two low-order bits are filled with 0s as shown in Figure 15-4.

Figure 15-4. μ-Law Transmit Data Companding Format



For transmission using A-law compression, the 13 data bits must be left-justified in DXR1, with the remaining three low-order bits filled with 0s as shown in Figure 15-5.

Figure 15-5. A-Law Transmit Data Companding Format



15.1.5.2 Capability to Compand Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. This can be used to:

- Convert linear to the appropriate μ-law or A-law format
- Convert μ-law or A-law to the linear format
- Observe the quantization effects in companding by transmitting linear data and compressing and re-expanding this data. This is useful only if both XCOMPAND and RCOMPAND enable the same companding format.

Figure 15-6 shows two methods by which the McBSP can compand internal data. Data paths for these two methods are used to indicate:

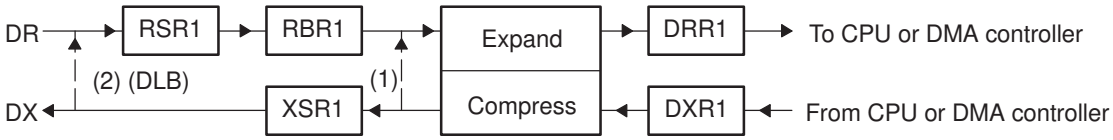
- When both the transmit and receive sections of the serial port are reset, DRR1 and DXR1 are connected internally through the companding logic. Values from DXR1 are compressed, as selected by XCOMPAND, and then expanded, as selected by RCOMPAND. RRDY and XRDY bits are not set. However, data is available in DRR1 within four CPU clocks after being written to DXR1.

  The advantage of this method is its speed. The disadvantage is that there is no synchronization available to the CPU and DMA to control the flow. DRR1 and DXR1 are internally connected if the (X/R)COMPAND bits are set to 10b or 11b (compand using μ-law or A-law).

Copyright © 2012–2019, Texas Instruments Incorporated

- The McBSP is enabled in digital loopback mode with companding appropriately enabled by RCOMPAND and XCOMPAND. Receive and transmit interrupts (RINT when RINTM = 0 and XINT when XINTM = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or DMA to these conversions, respectively. Here, the time for this companding depends on the serial bit rate selected.

Figure 15-6. Two Methods by Which the McBSP Can Compand Internal Data



15.1.5.3 Reversing Bit Order: Option to Transfer LSB First

Generally, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed during reception. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits, and LSB-first ordering is done.

15.2 Clocking and Framing Data

This section explains basic concepts and terminology important for understanding how McBSP data transfers are timed and delimited.

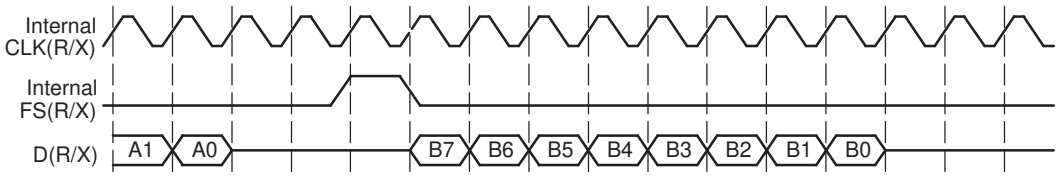
15.2.1 Clocking

Data is shifted one bit at a time from the DR pin to the RSR(s) or from the XSR(s) to the DX pin. The time for each bit transfer is controlled by the rising or falling edge of a clock signal.

The receive clock signal (CLKR) controls bit transfers from the DR pin to the RSR(s). The transmit clock signal (CLKX) controls bit transfers from the XSR(s) to the DX pin. CLKR or CLKX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP. The polarities of CLKR and CLKX are programmable.

In the example in Figure 15-7, the clock signal controls the timing of each bit transfer on the pin.

Figure 15-7. Example - Clock Signal Control of Bit Transfer Timing



NOTE: The McBSP cannot operate at a frequency faster than ½ the LSPCLK frequency. When driving CLKX or CLKR at the pin, choose an appropriate input clock frequency. When using the internal sample rate generator for CLKX and/or CLKR, choose an appropriate input clock frequency and divide down value (CLKDV) (i.e., be certain that CLKX or CLKR ≤ LSPCLK/2).

15.2.2 Serial Words

Bits traveling between a shift register (RSR or XSR) and a data pin (DR or DX) are transferred in a group called a serial word. You can define how many bits are in a word.

Copyright © 2012–2019, Texas Instruments Incorporated

Bits coming in on the DR pin are held in RSR until RSR holds a full serial word. Only then is the word passed to RBR (and ultimately to the DRR).

During transmission, XSR does not accept new data from DXR until a full serial word has been passed from XSR to the DX pin.

In the example in Figure 15-7, an 8-bit word size was defined (see bits 7 through 0 of word B being transferred).

15.2.3 Frames and Frame Synchronization

One or more words are transferred in a group called a frame. You can define how many words are in a frame.

All of the words in a frame are sent in a continuous stream. However, there can be pauses between frame transfers. The McBSP uses frame-synchronization signals to determine when each frame is received/transmitted. When a pulse occurs on a frame-synchronization signal, the McBSP begins receiving/transmitting a frame of data. When the next pulse occurs, the McBSP receives/transmits the next frame, and so on.

Pulses on the receive frame-synchronization (FSR) signal initiate frame transfers on DR. Pulses on the transmit frame-sync (FSX) signal initiate frame transfers on DX. FSR or FSX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP.

In the example in Figure 15-7, a one-word frame is transferred when a frame-synchronization pulse occurs.

In McBSP operation, the inactive-to-active transition of the frame-synchronization signal indicates the start of the next frame. For this reason, the frame-synchronization signal may be high for an arbitrary number of clock cycles. Only after the signal is recognized to have gone inactive, and then active again, does the next frame synchronization occur.

15.2.4 Generating Transmit and Receive Interrupts

The McBSP can send receive and transmit interrupts to the CPU to indicate specific events in the McBSP. To facilitate detection of frame synchronization, these interrupts can be sent in response to frame-synchronization pulses. Set the appropriate interrupt mode bits to 10b (for reception, RINTM = 10b; for transmission, XINTM = 10b).

15.2.4.1 Detecting Frame-Synchronization Pulses, Even in Reset State

Unlike other serial port interrupt modes, this mode can operate while the associated portion of the serial port is in reset (such as activating RINT when the receiver is in reset). In this case, FSRM/FSXM and FSRP/FSXP still select the appropriate source and polarity of frame synchronization. Thus, even when the serial port is in the reset state, these signals are synchronized to the CPU clock and then sent to the CPU in the form of RINT and XINT at the point at which they feed the receiver and transmitter of the serial port. Consequently, a new frame-synchronization pulse can be detected, and after this occurs the CPU can take the serial port out of reset safely.

15.2.5 Ignoring Frame-Synchronization Pulses

The McBSP can be configured to ignore transmit and/or receive frame-synchronization pulses. To have the receiver or transmitter recognize frame-synchronization pulses, clear the appropriate frame-synchronization ignore bit (RFIG = 0 for the receiver, XFIG = 0 for the transmitter). To have the receiver or transmitter ignore frame-synchronization pulses until the desired frame length or number of words is reached, set the appropriate frame-synchronization ignore bit (RFIG = 1 for the receiver, XFIG = 1 for the transmitter). For more details on unexpected frame-synchronization pulses, see one of the following topics:

- *Unexpected Receive Frame-Synchronization Pulse* (see Section 15.5.3)
- *Unexpected Transmit Frame-Synchronization Pulse* (see Section 15.5.5)

You can also use the frame-synchronization ignore function for data packing (for more details, see Section 15.11.2).

Copyright © 2012–2019, Texas Instruments Incorporated

15.2.6 Frame Frequency

The frame frequency is determined by the period between frame-synchronization pulses and is defined as shown by Example 1.

Equation 1: McBSP Frame Frequency

$$\text{Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Clock Cycles Between Frame- Sync Pulses}}$$

The frame frequency can be increased by decreasing the time between frame-synchronization pulses (limited only by the number of bits per frame). As the frame transmit frequency increases, the inactivity period between the data packets for adjacent transfers decreases to zero.

15.2.7 Maximum Frame Frequency

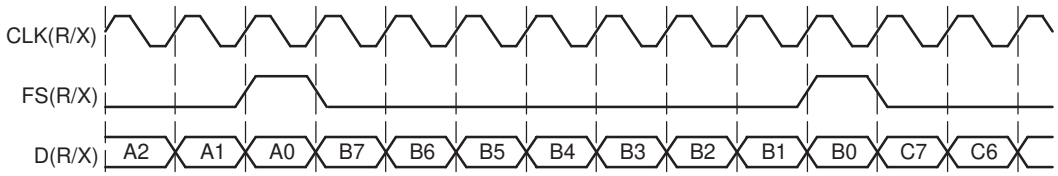
The minimum number of clock cycles between frame synchronization pulses is equal to the number of bits transferred per frame. The maximum frame frequency is defined as shown by Example 2.

Equation 2: McBSP Maximum Frame Frequency

$$\text{Maximum Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Bits Per Frame}}$$

Figure 15-8 shows the McBSP operating at maximum packet frequency. At maximum packet frequency, the data bits in consecutive packets are transmitted contiguously with no inactivity between bits.

Figure 15-8. McBSP Operating at Maximum Packet Frequency



If there is a 1-bit data delay as shown in this figure, the frame-synchronization pulse overlaps the last bit transmitted in the previous frame. Effectively, this permits a continuous stream of data, making frame-synchronization pulses redundant. Theoretically, only an initial frame-synchronization pulse is required to initiate a multipacket transfer.

The McBSP supports operation of the serial port in this fashion by ignoring the successive frame-synchronization pulses. Data is clocked into the receiver or clocked out of the transmitter during every clock cycle.

NOTE: For XDATDLY = 0 (0-bit data delay), the first bit of data is transmitted asynchronously to the internal transmit clock signal (CLKX). For more details, see Section 15.9.12, *Set the Transmit Data Delay*.

15.3 Frame Phases

The McBSP allows you to configure each frame to contain one or two phases. The number of words and the number of bits per word can be specified differently for each of the two phases of a frame, allowing greater flexibility in structuring data transfers. For example, you might define a frame as consisting of one phase containing two words of 16 bits each, followed by a second phase consisting of 10 words of 8 bits each. This configuration permits you to compose frames for custom applications or, in general, to maximize the efficiency of data transfers.

Copyright © 2012–2019, Texas Instruments Incorporated

15.3.1 Number of Phases, Words, and Bits Per Frame

Table 15-2 shows which bit-fields in the receive control registers (RCR1 and RCR2) and in the transmit control registers (XCR1 and XCR2) determine the number of phases per frame, the number of words per frame, and number of bits per word for each phase, for the receiver and transmitter. The maximum number of words per frame is 128 for a single-phase frame and 256 for a dual-phase frame. The number of bits per word can be 8, 12, 16, 20, 24, or 32 bits.

Table 15-2. Register Bits That Determine the Number of Phases, Words, and Bits

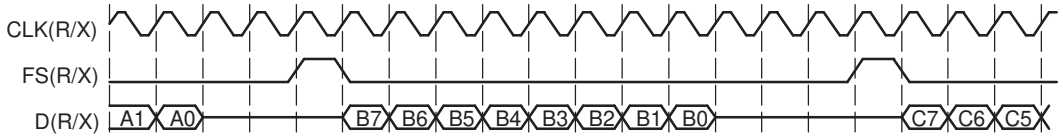
Operation	Number of Phases	Words per Frame Set With	Bits per Word Set With
Reception	1 (RPHASE = 0)	RFRLEN1	RWDLEN1
Reception	2 (RPHASE = 1)	RFRLEN1 and RFRLEN2	RWDLEN1 for phase 1
			RWDLEN2 for phase 2
Transmission	1 (XPHASE = 0)	XFRLEN1	XWDLEN1
Transmission	2 (XPHASE = 1)	XFRLEN1 and XFRLEN2	XWDLEN1 for phase 1
			XWDLEN2 for phase 2

15.3.2 Single-Phase Frame Example

Figure 15-9 shows an example of a single-phase data frame containing one 8-bit word. Because the transfer is configured for one data bit delay, the data on the DX and DR pins are available one clock cycle after FS(R/X) goes active. The figure makes the following assumptions:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0b: 1 word per frame
- (R/X)WDLEN1 = 000b: 8-bit word length
- (R/X)FRLEN2 and (R/X)WDLEN2 are ignored
- CLK(X/R)P = 0: Receive data clocked on falling edge; transmit data clocked on rising edge
- FS(R/X)P = 0: Active-high frame-synchronization signals
- (R/X)DATDLY = 01b: 1-bit data delay

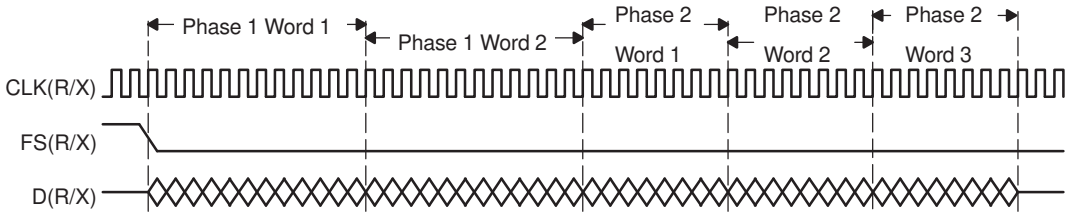
Figure 15-9. Single-Phase Frame for a McBSP Data Transfer



15.3.3 Dual-Phase Frame Example

Figure 15-10 shows an example of a frame where the first phase consists of two words of 12 bits each, followed by a second phase of three words of 8 bits each. The entire bit stream in the frame is contiguous. There are no gaps either between words or between phases.

Figure 15-10. Dual-Phase Frame for a McBSP Data Transfer



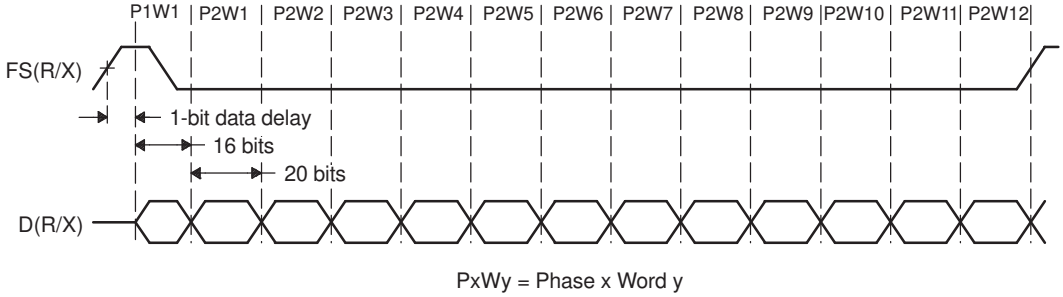
A XRDY gets asserted once per phase. So, if there are 2 phases, XRDY gets asserted twice (once per phase).

Copyright © 2012–2019, Texas Instruments Incorporated

15.3.4 Implementing the AC97 Standard With a Dual-Phase Frame

Figure 15-11 shows an example of the Audio Codec '97 (AC97) standard, which uses the dual-phase frame feature. Notice that words, not individual bits, are shown on the D(R/X) signal. The first phase (P1) consists of a single 16-bit word. The second phase (P2) consists of twelve 20-bit words. The phase configurations are listed after the figure.

Figure 15-11. Implementing the AC97 Standard With a Dual-Phase Frame

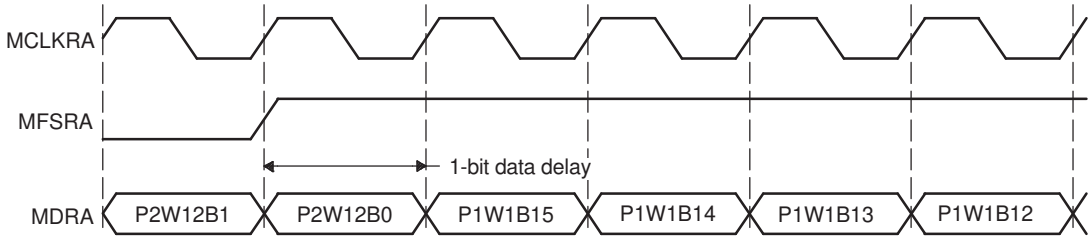


PxWy = Phase x Word y

- (R/X)PHASE = 1: Dual-phase frame
- (R/X)FRLEN1 = 0000000b: 1 word in phase 1
- (R/X)WDLEN1 = 010b: 16 bits per word in phase 1
- (R/X)FRLEN2 = 0001011b: 12 words in phase 2
- (R/X)WDLEN2 = 011b: 20 bits per word in phase 2
- CLKRP/CLKXP= 0: Receive data sampled on falling edge of internal CLKR / transmit data clocked on rising edge of internal CLKX
- FSRP/FSXP = 0: Active-high frame-sync signal
- (R/X)DATDLY = 01b: Data delay of 1 clock cycle (1-bit data delay)

Figure 15-12 shows the timing of an AC97-standard data transfer near frame synchronization. In this figure, individual bits are shown on D(R/X). Specifically, the figure shows the last two bits of phase 2 of one frame and the first four bits of phase 1 of the next frame. Regardless of the data delay, data transfers can occur without gaps. The first bit of the second frame (P1W1B15) immediately follows the last bit of the first frame (P2W12B0). Because a 1-bit data delay has been chosen, the transition on the frame-sync signal can occur when P2W12B0 is transferred.

Figure 15-12. Timing of an AC97-Standard Data Transfer Near Frame Synchronization



PxWyBz = Phase x Word y Bit z

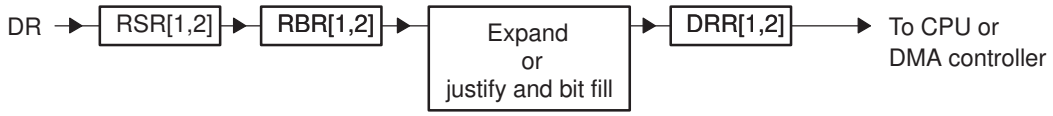
15.3.5 McBSP Reception

This section explains the fundamental process of reception in the McBSP. For details about how to program the McBSP receiver, see *Receiver Configuration* in Section 15.8.

Figure 15-13 and Figure 15-14 show how reception occurs in the McBSP. Figure 15-13 shows the physical path for the data. Figure 15-14 is a timing diagram showing signal activity for one possible reception scenario. A description of the process follows the figures.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-13. McBSP Reception Physical Data Path

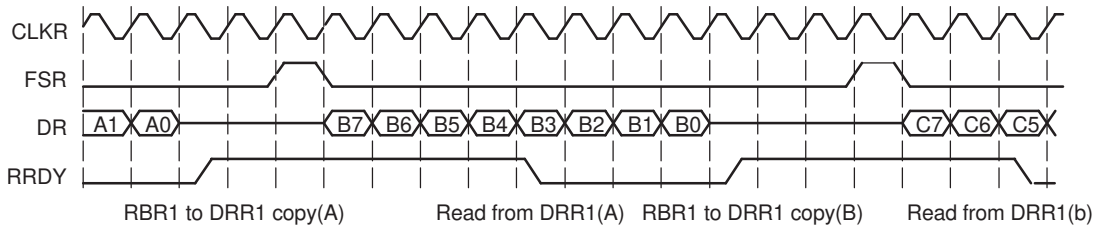


A RSR[1,2]: Receive shift registers 1 and 2

B RBR[1,2]: Receive buffer registers 1 and 2

C DRR[1,2]: Data receive registers 1 and 2

Figure 15-14. McBSP Reception Signal Activity



A CLKR: Internal receive clock

B FSR: Internal receive frame-synchronization signal

C DR: Data on DR pin

D RRDY: Status of receiver ready bit (high is 1)

The following process describes how data travels from the DR pin to the CPU or to the DMA controller:

1. The McBSP waits for a receive frame-synchronization pulse on internal FSR.
2. When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the RDATDLY bits of RCR2.

   In the preceding timing diagram (Figure 15-14), a 1-bit data delay is selected.
3. The McBSP accepts data bits on the DR pin and shifts them into the receive shift register(s).

   If the word length is 16 bits or smaller, only RSR1 is used. If the word length is larger than 16 bits, RSR2 and RSR1 are used and RSR2 contains the most significant bits. For details on choosing a word length, see Section 15.8.8, *Set the Receive Word Length(s)*.
4. When a full word is received, the McBSP copies the contents of the receive shift register(s) to the receive buffer register(s), provided that RBR1 is not full with previous data.

   If the word length is 16 bits or smaller, only RBR1 is used. If the word length is larger than 16 bits, RBR2 and RBR1 are used and RBR2 contains the most significant bits.
5. The McBSP copies the contents of the receive buffer register(s) into the data receive register(s), provided that DRR1 is not full with previous data. When DRR1 receives new data, the receiver ready bit (RRDY) is set in SPCR1. This indicates that received data is ready to be read by the CPU or the DMA controller.

   If the word length is 16 bits or smaller, only DRR1 is used. If the word length is larger than 16 bits, DRR2 and DRR1 are used and DRR2 contains the most significant bits.

   If companding is used during the copy (RCOMPAND = 10b or 11b in RCR2), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DRR1. If companding is disabled, the data copied from RBR[1,2] to DRR[1,2] is justified and bit filled according to the RJUST bits.
6. The CPU or the DMA controller reads the data from the data receive register(s). When DRR1 is read, RRDY is cleared and the next RBR-to-DRR copy is initiated.

NOTE: If both DRRs are required (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

When activity is not properly timed, errors can occur. See the following topics for more details:

Copyright © 2012–2019, Texas Instruments Incorporated

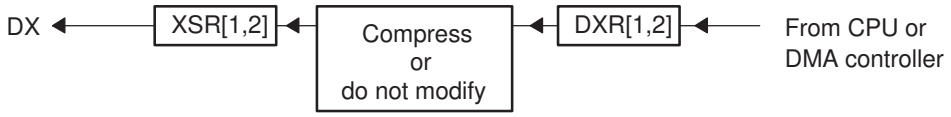
- *Overrun in the Receiver* (see Section 15.5.2)
- *Unexpected Receive Frame-Synchronization Pulse* (see Section 15.5.3)

15.3.6 McBSP Transmission

This section explains the fundamental process of transmission in the McBSP. For details about how to program the McBSP transmitter, see Section 15.9, *Transmitter Configuration*.

Figure 15-15 and Figure 15-16 show how transmission occurs in the McBSP. Figure 15-15 shows the physical path for the data. Figure 15-16 is a timing diagram showing signal activity for one possible transmission scenario. A description of the process follows the figures.

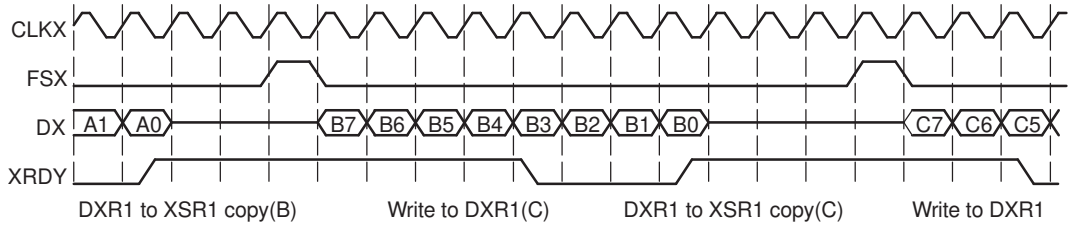
Figure 15-15. McBSP Transmission Physical Data Path



A XSR[1,2]: Transmit shift registers 1 and 2

B DXR[1,2]: Data transmit registers 1 and 2

Figure 15-16. McBSP Transmission Signal Activity



A CLKX: Internal transmit clock

B FSX: Internal transmit frame-synchronization signal

C DX: Data on DX pin

D XRDY: Status of transmitter ready bit (high is 1)

1. The CPU or the DMA controller writes data to the data transmit register(s). When DXR1 is loaded, the transmitter ready bit (XRDY) is cleared in SPCR2 to indicate that the transmitter is not ready for new data.

   If the word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, DXR2 and DXR1 are used and DXR2 contains the most significant bits. For details on choosing a word length, see Section 15.9.8, *Set the Transmit Word Length(s)*.

   NOTE: If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

2. When new data arrives in DXR1, the McBSP copies the content of the data transmit register(s) to the transmit shift register(s). In addition, the transmit ready bit (XRDY) is set. This indicates that the transmitter is ready to accept new data from the CPU or the DMA controller.

   If the word length is 16 bits or smaller, only XSR1 is used. If the word length is larger than 16 bits, XSR2 and XSR1 are used and XSR2 contains the most significant bits.

   If companding is used during the transfer (XCOMPAND = 10b or 11b in XCR2), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in the μ-law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXR(s) to the XSR(s) without modification.

3. The McBSP waits for a transmit frame-synchronization pulse on internal FSX.

4. When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the XDATDLY bits of XCR2.

Copyright © 2012–2019, Texas Instruments Incorporated

In the preceding timing diagram (Figure 15-16), a 1-bit data delay is selected.

5. The McBSP shifts data bits from the transmit shift register(s) to the DX pin.

When activity is not properly timed, errors can occur. See the following topics for more details:

- *Overwrite in the Transmitter* (Section 15.5.4)
- *Underflow in the Transmitter* (Section 15.5.4.3)
- *Unexpected Transmit Frame-Synchronization Pulse* (Section 15.5.5)

15.3.7 Interrupts and DMA Events Generated by a McBSP

The McBSP sends notification of important events to the CPU and the DMA controller via the internal signals shown in Table 15-3.

Table 15-3. Interrupts and DMA Events Generated by a McBSP

Internal Signal	Description
RINT	Receive interrupt
	The McBSP can send a receive interrupt request to CPU based upon a selected condition in the receiver of the McBSP (a condition selected by the RINTM bits of SPCR1).
XINT	Transmit interrupt
	The McBSP can send a transmit interrupt request to CPU based upon a selected condition in the transmitter of the McBSP (a condition selected by the XINTM bits of SPCR2).
REVT	Receive synchronization event
	An REVT signal is sent to the DMA when data has been received in the data receive registers (DRRs).
XEVT	Transmit synchronization event
	An XEVT signal is sent to the DMA when the data transmit registers (DXRs) are ready to accept the next serial word for transmission.

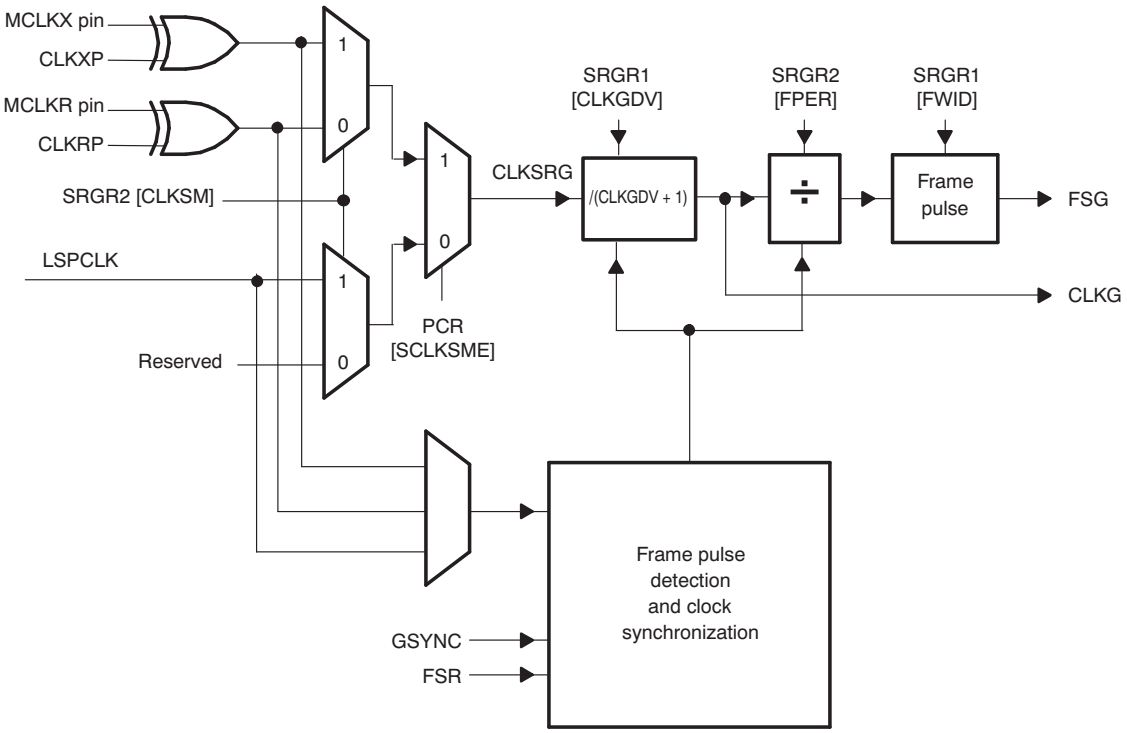
15.4 McBSP Sample Rate Generator

Each McBSP contains a sample rate generator (SRG) that can be programmed to generate an internal data clock (CLKG) and an internal frame-synchronization signal (FSG). CLKG can be used for bit shifting on the data receive (DR) pin and/or the data transmit (DX) pin. FSG can be used to initiate frame transfers on DR and/or DX. Figure 15-17 is a conceptual block diagram of the sample rate generator.

Copyright © 2012–2019, Texas Instruments Incorporated

15.4.1 Block Diagram

Figure 15-17. Conceptual Block Diagram of the Sample Rate Generator



The source clock for the sample rate generator (labeled CLKSRG in the diagram) can be supplied by the LSPCLK, or by an external pin (MCLKX or MCLKR). The source is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2. If a pin is used, the polarity of the incoming signal can be inverted with the appropriate polarity bit (CLKXP of PCR or CLKRP of PCR).

The sample rate generator has a three-stage clock divider that gives CLKG and FSG programmability. The three stages provide:

- Clock divide-down. The source clock is divided according to the CLKGDV bits of SRGR1 to produce CLKG.
- Frame period divide-down. CLKG is divided according to the FPER bits of SRGR2 to control the period from the start of a frame-pulse to the start of the next pulse.
- Frame-synchronization pulse-width countdown. CLKG cycles are counted according to the FWID bits of SRGR1 to control the width of each frame-synchronization pulse.

NOTE: The McBSP cannot operate at a frequency faster than ½ the source clock frequency. Choose an input clock frequency and a CLKGDV value such that CLKG is less than or equal to ½ the source clock frequency.

In addition to the three-stage clock divider, the sample rate generator has a frame-synchronization pulse detection and clock synchronization module that allows synchronization of the clock divide down with an incoming frame-synchronization pulse on the FSR pin. This feature is enabled or disabled with the GSYNC bit of SRGR2.

For details on getting the sample rate generator ready for operation, see Section 15.4.4, *Reset and Initialization Procedure*.

Copyright © 2012–2019, Texas Instruments Incorporated

15.4.1.1 Clock Generation in the Sample Rate Generator

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both. Use of the sample rate generator to drive clocking is controlled by the clock mode bits (CLKRM and CLKXM) in the pin control register (PCR). When a clock mode bit is set to 1 (CLKRM = 1 for reception, CLKXM = 1 for transmission), the corresponding data clock (CLKR for reception, CLKX for transmission) is driven by the internal sample rate generator output clock (CLKG).

The effects of CLKRM = 1 and CLKXM = 1 on the McBSP are partially affected by the use of the digital loopback mode and the clock stop (SPI) mode, respectively, as described in Table 15-4. The digital loopback mode (described in Section 15.8.4) is selected with the DLB bit of SPCR1. The clock stop mode (described in Section 15.7.2) is selected with the CLKSTP bits of SPCR1.

When using the sample rate generator as a clock source, make sure the sample rate generator is enabled (GRST = 1).

Table 15-4. Effects of DLB and CLKSTP on Clock Modes

Mode Bit Settings		Effect
CLKRM = 1	DLB = 0 (Digital loopback mode disabled)	CLKR is an output pin driven by the sample rate generator output clock (CLKG).
	DLB = 1 (Digital loopback mode enabled)	CLKR is an output pin driven by internal CLKX. The source for CLKX depends on the CLKXM bit.
CLKXM = 1	CLKSTP = 00b or 01b (Clock stop (SPI) mode disabled)	CLKX is an output pin driven by the sample rate generator output clock (CLKG).
	CLKSTP = 10b or 11b (Clock stop (SPI) mode enabled)	The McBSP is a master in an SPI system. Internal CLKX drives internal CLKR and the shift clocks of any SPI-compliant slave devices in the system. CLKX is driven by the internal sample rate generator.

15.4.1.2 Choosing an Input Clock

The sample rate generator must be driven by an input clock signal from one of the three sources selectable with the SCLKME bit of PCR and the CLKSM bit of SRGR2 (see Table 15-5). When CLKSM = 1, the minimum divide down value in CLKGDV bits is 1. CLKGDV is described in Section 15.4.1.4.

Table 15-5. Choosing an Input Clock for the Sample Rate Generator with the SCLKME and CLKSM Bits

SCLKME	CLKSM	Input Clock for Sample Rate Generator
0	0	Reserved
0	1	LSPCLK
1	0	Signal on MCLKR pin
1	1	Signal on MCLKX pin

15.4.1.3 Choosing a Polarity for the Input Clock

As shown in Figure 15-18, when the input clock is received from a pin, you can choose the polarity of the input clock. The rising edge of CLKSRG generates CLKG and FSG, but you can determine which edge of the input clock causes a rising edge on CLKSRG. The polarity options and their effects are described in Table 15-6.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-18. Possible Inputs to the Sample Rate Generator and the Polarity Bits

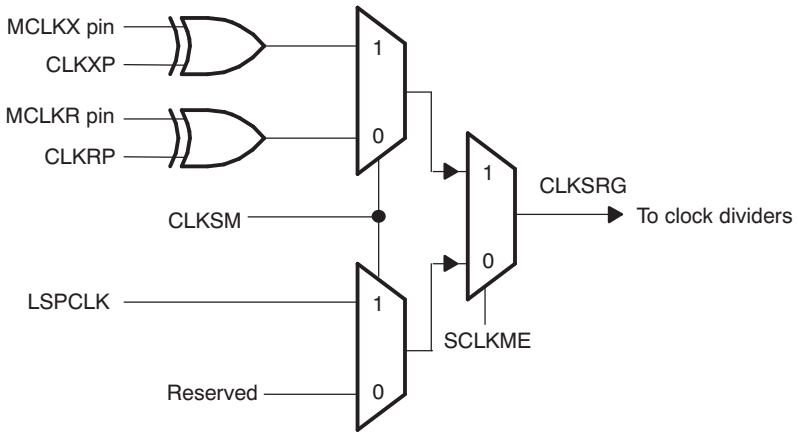


Table 15-6. Polarity Options for the Input to the Sample Rate Generator

Input Clock	Polarity Option	Effect
LSPCLK	Always positive polarity	Rising edge of CPU clock generates transitions on CLKG and FSG.
Signal on MCLKR pin	CLKRP = 0 in PCR	Falling edge on MCLKR pin generates transitions on CLKG and FSG.
	CLKRP = 1 in PCR	Rising edge on MCLKR pin generates transitions on CLKG and FSG.
Signal on MCLKX pin	CLKXP = 0 in PCR	Rising edge on MCLKX pin generates transitions on CLKG and FSG.
	CLKXP = 1 in PCR	Falling edge on MCLKX pin generates transitions on CLKG and FSG.

15.4.1.4 Choosing a Frequency for the Output Clock (CLKG)

The input clock (LSPCLK or external clock) can be divided down by a programmable value to drive CLKG. Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see Figure 15-17) generates CLKG and FSG.

The first divider stage of the sample rate generator creates the output clock from the input clock. This divider stage uses a counter that is preloaded with the divide down value in the CLKGDV bits of SRGR1. The output of this stage is the data clock (CLKG). CLKG has the frequency represented by Example 3.

Equation 3: CLKG Frequency

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

Thus, the input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide down, the high-state duration is p+1 cycles and the low-state duration is p cycles.

15.4.1.5 Keeping CLKG Synchronized to External MCLKR

When the MCLKR pin is used to drive the sample rate generator (see Section 15.4.1.2), the GSYNC bit in SRGR2 and the FSR pin can be used to configure the timing of the output clock (CLKG) relative to the input clock. Note that this feature is available only when the MCLKR pin is used to feed the external clock.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

For more details about synchronization, see Section 15.4.3.

Copyright © 2012–2019, Texas Instruments Incorporated

15.4.2 Frame Synchronization Generation in the Sample Rate Generator

The sample rate generator can produce a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both.

If you want the receiver to use FSG for frame synchronization, make sure FSRM = 1. (When FSRM = 0, receive frame synchronization is supplied via the FSR pin.)

If you want the transmitter to use FSG for frame synchronization, you must set:

- FSXM = 1 in PCR: This indicates that transmit frame synchronization is supplied by the McBSP itself rather than from the FSX pin.
- FSGM = 1 in SRGR2: This indicates that when FSXM = 1, transmit frame synchronization is supplied by the sample rate generator. (When FSGM = 0 and FSXM = 1, the transmitter uses frame-synchronization pulses generated every time data is transferred from DXR[1,2] to XSR[1,2].)

In either case, the sample rate generator must be enabled (GRST = 1) and the frame-synchronization logic in the sample rate generator must be enabled (FRST = 0).

15.4.2.1 Choosing the Width of the Frame-Synchronization Pulse on FSG

Each pulse on FSG has a programmable width. You program the FWID bits of SRGR1, and the resulting pulse width is (FWID + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

15.4.2.2 Controlling the Period Between the Starting Edges of Frame-Synchronization Pulses on FSG

You can control the amount of time from the starting edge of one FSG pulse to the starting edge of the next FSG pulse. This period is controlled in one of two ways, depending on the configuration of the sample rate generator:

- If the sample rate generator is using an external input clock and GSYNC = 1 in SRGR2, FSG pulses in response to an inactive-to-active transition on the FSR pin. Thus, the frame-synchronization period is controlled by an external device.
- Otherwise, you program the FPER bits of SRGR2, and the resulting frame-synchronization period is (FPER + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

15.4.2.3 Keeping FSG Synchronized to an External Clock

When an external signal is selected to drive the sample rate generator (see Section 15.4.1.2 on page Section 15.4.1.2), the GSYNC bit of SRGR2 and the FSR pin can be used to configure the timing of FSG pulses.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

See Section 15.4.3 for more details about synchronization.

15.4.3 Synchronizing Sample Rate Generator Outputs to an External Clock

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) based on an input clock signal that is either the CPU clock signal or a signal at the MCLKR or MCLKX pin. When an external clock is selected to drive the sample rate generator, the GSYNC bit of SRGR2 and the FSR pin can be used to control the timing of CLKG and the pulsing of FSG relative to the chosen input clock.

Make GSYNC = 1 when you want the McBSP and an external device to divide down the input clock with the same phase relationship. If GSYNC = 1:

- An inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and a pulsing of FSG.
- CLKG always begins with a high state after synchronization.
- FSR is always detected at the same edge of the input clock signal that generates CLKG, no matter how long the FSR pulse is.

Copyright © 2012–2019, Texas Instruments Incorporated

- The FPER bits of SRGR2 are ignored because the frame-synchronization period on FSG is determined by the arrival of the next frame-synchronization pulse on the FSR pin.

If GSYNC = 0, CLKG runs freely and is not resynchronized, and the frame-synchronization period on FSG is determined by FPER.

15.4.3.1 Operating the Transmitter Synchronously with the Receiver

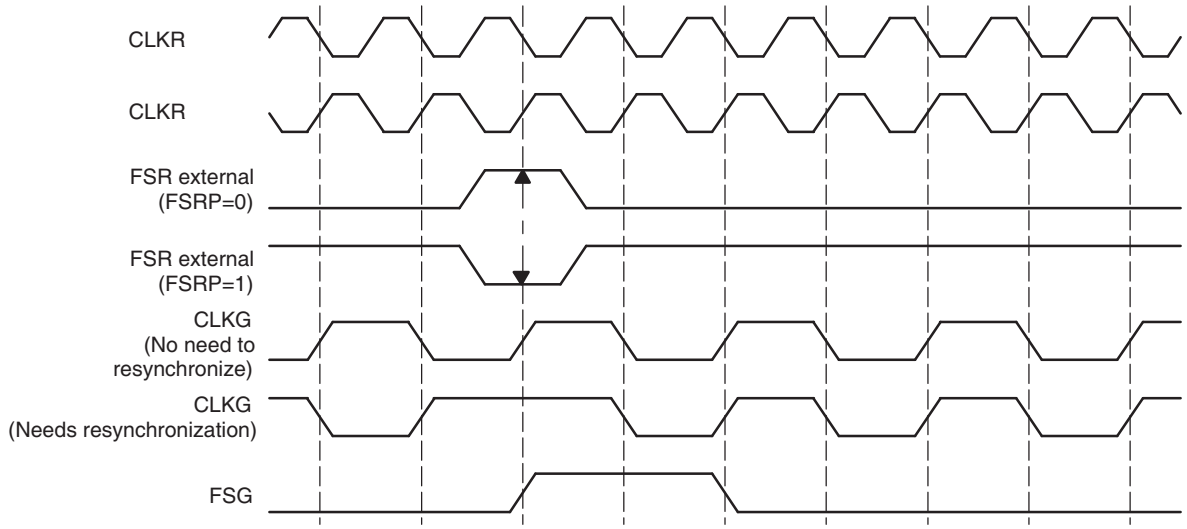
When GSYNC = 1, the transmitter can operate synchronously with the receiver, provided that:

- FSX is programmed to be driven by FSG (FSGM = 1 in SRGR2 and FSXM = 1 in PCR). If the input FSR has appropriate timing so that it can be sampled by the falling edge of CLKG, it can be used, instead, by setting FSXM = 0 and connecting FSR to FSX externally.
- The sample rate generator clock drives the transmit and receive clocking (CLKRM = CLKXM = 1 in PCR).

15.4.3.2 Synchronization Examples

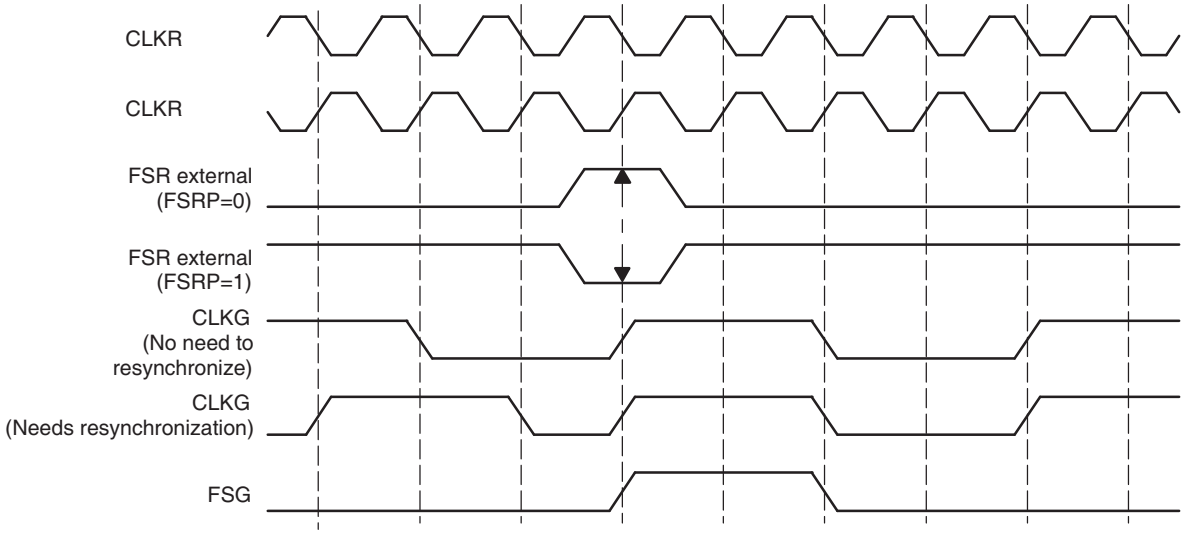
Figure 15-19 and Figure 15-20 show the clock and frame-synchronization operation with various polarities of CLKR and FSR. These figures assume FWID = 0 in SRGR1, for an FSG pulse that is one CLKG cycle wide. The FPER bits of SRGR2 are not programmed; the period from the start of a frame-synchronization pulse to the start of the next pulse is determined by the arrival of the next inactive-to-active transition on the FSR pin. Each of the figures shows what happens to CLKG when it is initially synchronized and GSYNC = 1, and when it is not initially synchronized and GSYNC = 1. The second figure has a slower CLKG frequency (it has a larger divide-down value in the CLKGDV bits of SRGR1).

Figure 15-19. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-20. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3



15.4.4 Reset and Initialization Procedure for the Sample Rate Generator

To reset and initialize the sample rate generator:

Step 1. Place the McBSP/sample rate generator in reset.

During a DSP reset, the sample rate generator, the receiver, and the transmitter reset bits (GRST, RRST, and XRST) are automatically forced to 0. Otherwise, during normal operation, the sample rate generator can be reset by making GRST = 0 in SPCR2, provided that CLKG and/or FSG is not used by any portion of the McBSP. Depending on your system you may also want to reset the receiver (RRST = 0 in SPCR1) and reset the transmitter (XRST = 0 in SPCR2).

If GRST = 0 due to a device reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven inactive-low. If GRST = 0 due to program code, CLKG and FSG are driven low (inactive).

Step 2. Program the registers that affect the sample rate generator.

Program the sample rate generator registers (SRGR1 and SRGR2) as required for your application. If necessary, other control registers can be loaded with desired values, provided the respective portion of the McBSP (the receiver or transmitter) is in reset.

After the sample rate generator registers are programmed, wait 2 CLKSRG cycles. This ensures proper synchronization internally.

Step 3. Enable the sample rate generator (take it out of reset).

In SPCR2, make GRST = 1 to enable the sample rate generator.

After the sample rate generator is enabled, wait two CLKG cycles for the sample rate generator logic to stabilize.

On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to Example 4.

Table 15-7. Input Clock Selection for Sample Rate Generator

SCLKME	CLKSM	Input Clock for Sample Rate Generator
0	0	Reserved
0	1	LSPCLK
1	0	Signal on MCLKR pin
1	1	Signal on MCLKX pin

Copyright © 2012–2019, Texas Instruments Incorporated

Step 4. If necessary, enable the receiver and/or the transmitter.

If necessary, remove the receiver and/or transmitter from reset by setting RRST and/or XRST = 1.

Step 5. If necessary, enable the frame-synchronization logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1,2] is loaded with data), set GRST = 1 in SPCR2 if an internally generated frame-synchronization pulse is required. FSG is generated with an active-high edge after the programmed number of CLKG clocks (FPER + 1) have elapsed.

Equation 4: CLKG Frequency

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

where the input clock is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2 in one of the configurations shown in Table 15-7.

15.5 McBSP Exception/Error Conditions

This section describes exception/error conditions and how to handle them.

15.5.1 Types of Errors

There are five serial port events that can constitute a system error:

- Receiver overrun (RFULL = 1)

  This occurs when DRR1 has not been read since the last RBR-to-DRR copy. Consequently, the receiver does not copy a new word from the RBR(s) to the DRR(s) and the RSR(s) are now full with another new word shifted in from DR. Therefore, RFULL = 1 indicates an error condition wherein any new data that can arrive at this time on DR replaces the contents of the RSR(s), and the previous word is lost. The RSRs continue to be overwritten as long as new data arrives on DR and DRR1 is not read. For more details about overrun in the receiver, see Section 15.5.2.
- Unexpected receive frame-synchronization pulse (RSYNCERR = 1)

  This occurs during reception when RFIG = 0 and an unexpected frame-synchronization pulse occurs. An unexpected frame-synchronization pulse is one that begins the next frame transfer before all the bits of the current frame have been received. Such a pulse causes data reception to abort and restart. If new data has been copied into the RBR(s) from the RSR(s) since the last RBR-to-DRR copy, this new data in the RBR(s) is lost. This is because no RBR-to-DRR copy occurs; the reception has been restarted. For more details about receive frame-synchronization errors, see Section 15.5.3.
- Transmitter data overwrite

  This occurs when the CPU or DMA controller overwrites data in the DXR(s) before the data is copied to the XSR(s). The overwritten data never reaches the DX pin. For more details about overwrite in the transmitter, see Section 15.5.4.
- Transmitter underflow ($\overline{\text{XEMPTY}}$ = 0)

  If a new frame-synchronization signal arrives before new data is loaded into DXR1, the previous data in the DXR(s) is sent again. This procedure continues for every new frame-synchronization pulse that arrives until DXR1 is loaded with new data. For more details about underflow in the transmitter, see Section 15.5.4.3.
- Unexpected transmit frame-synchronization pulse (XSYNCERR = 1)

  This occurs during transmission when XFIG = 0 and an unexpected frame-synchronization pulse occurs. An unexpected frame-synchronization pulse is one that begins the next frame transfer before all the bits of the current frame have been transferred. Such a pulse causes the current data transmission to abort and restart. If new data has been written to the DXR(s) since the last DXR-to-XSR copy, the current value in the XSR(s) is lost. For more details about transmit frame-synchronization errors, see Section 15.5.5.

15.5.2 Overrun in the Receiver

RFULL = 1 in SPCR1 indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when all of the following conditions are met:

Copyright © 2012–2019, Texas Instruments Incorporated

1. DRR1 has not been read since the last RBR-to-DRR copy (RRDY = 1).
2. RBR1 is full and an RBR-to-DRR copy has not occurred.
3. RSR1 is full and an RSR1-to-RBR copy has not occurred.

As described in the Section 15.3.5, *McBSP Reception*, data arriving on DR is continuously shifted into RSR1 (for word length of 16 bits or smaller) or RSR2 and RSR1 (for word length larger than 16 bits). Once a complete word is shifted into the RSR(s), an RSR-to-RBR copy can occur only if the previous data in RBR1 has been copied to DRR1. The RRDY bit is set when new data arrives in DRR1 and is cleared when that data is read from DRR1. Until RRDY = 0, the next RBR-to-DRR copy does not take place, and the data is held in the RSR(s). New data arriving on the DR pin is shifted into RSR(s), and the previous content of the RSR(s) is lost.

You can prevent the loss of data if DRR1 is read no later than 2.5 cycles before the end of the third word is shifted into the RSR1.

NOTE: If both DRRs are needed (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

After the receiver starts running from reset, a minimum of three words must be received before RFULL is set. Either of the following events clears the RFULL bit and allows subsequent transfers to be read properly:

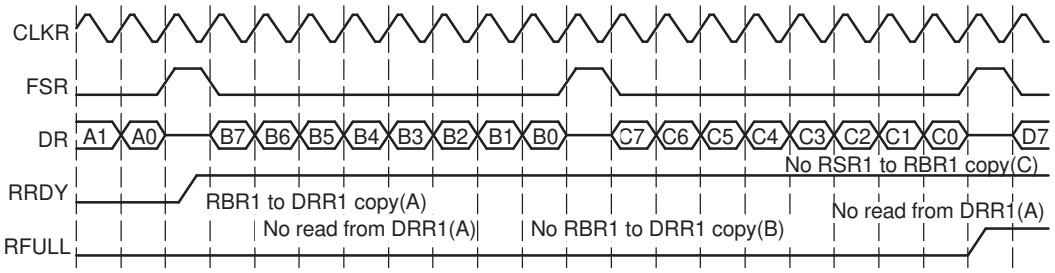
- The CPU or DMA controller reads DRR1.
- The receiver is reset individually (RRST = 0) or as part of a device reset.

Another frame-synchronization pulse is required to restart the receiver.

15.5.2.1 Example of Overrun Condition

Figure 15-21 shows the receive overrun condition. Because serial word A is not read from DRR1 before serial word B arrives in RBR1, B is not transferred to DRR1 yet. Another new word ©) arrives and RSR1 is full with this data. DRR1 is finally read, but not earlier than 2.5 cycles before the end of word C. Therefore, new data (D) overwrites word C in RSR1. If DRR1 is not read in time, the next word can overwrite D.

Figure 15-21. Overrun in the McBSP Receiver

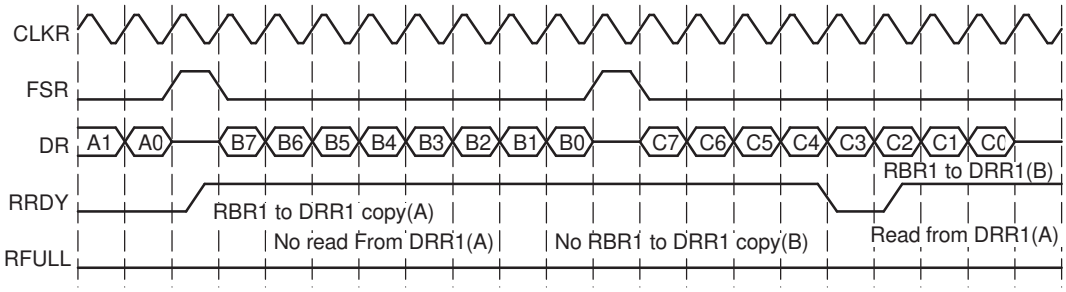


15.5.2.2 Example of Preventing Overrun Condition

Figure 15-22 shows the case where RFULL is set, but the overrun condition is prevented by a read from DRR1 at least 2.5 cycles before the next serial word ©) is completely shifted into RSR1. This ensures that an RBR1-to-DRR1 copy of word B occurs before receiver attempts to transfer word C from RSR1 to RBR1.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-22. Overrun Prevented in the McBSP Receiver



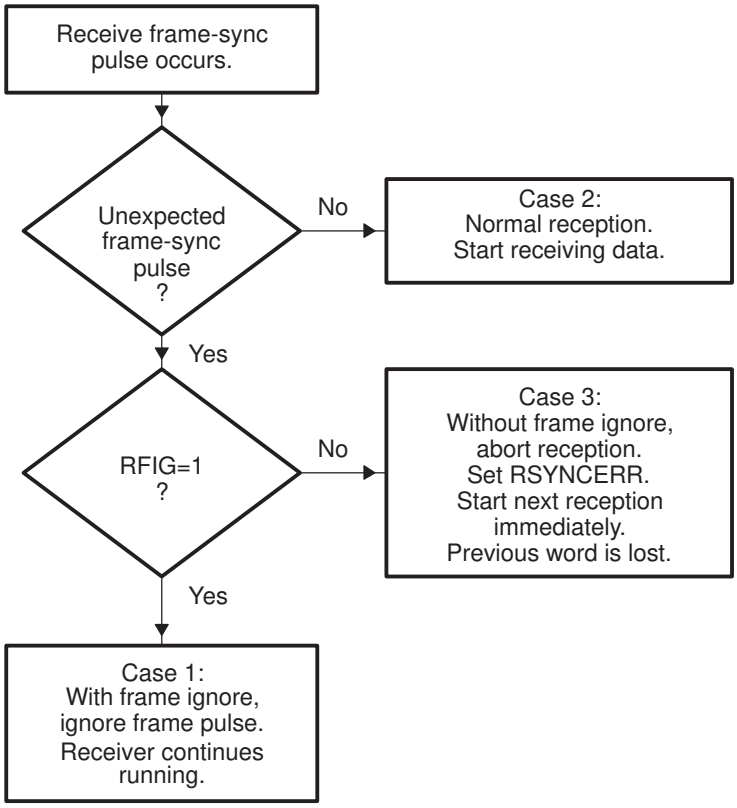
15.5.3 Unexpected Receive Frame-Synchronization Pulse

Section 15.5.3.1 shows how the McBSP responds to any receive frame-synchronization pulses, including an unexpected pulse. Section 15.5.3.2 and Section 15.5.3.3 show an examples of a frame-synchronization error and an example of how to prevent such an error, respectively.

15.5.3.1 Possible Responses to Receive Frame-Synchronization Pulses

Figure 15-23 shows the decision tree that the receiver uses to handle all incoming frame-synchronization pulses. The figure assumes that the receiver has been started (RRST = 1 in SPCR1). Case 3 in the figure is the case in which an error occurs.

Figure 15-23. Possible Responses to Receive Frame-Synchronization Pulses



Any one of three cases can occur:

- Case 1: Unexpected internal FSR pulses with RFIG = 1 in RCR2. Receive frame-synchronization pulses are ignored, and the reception continues.
- Case 2: Normal serial port reception. Reception continues normally because the frame-synchronization

Copyright © 2012–2019, Texas Instruments Incorporated

pulse is not unexpected. There are three possible reasons why a receive operation might *not* be in progress when the pulse occurs:

- The FSR pulse is the first after the receiver is enabled (RRST = 1 in SPCR1).
- The FSR pulse is the first after DRR[1,2] is read, clearing a receiver full (RFULL = 1 in SPCR1) condition.
- The serial port is in the interpacket intervals. The programmed data delay for reception (programmed with the RDATDLY bits in RCR2) may start during these interpacket intervals for the first bit of the next word to be received. Thus, at maximum frame frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.

- Case 3: Unexpected receive frame synchronization with RFIG = 0 (frame-synchronization pulses not ignored). Unexpected frame-synchronization pulses can originate from an external source or from the internal sample rate generator.

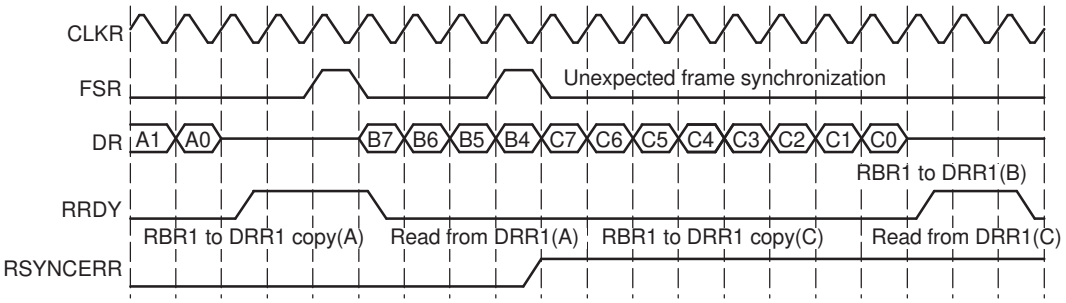
  If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse, and the receiver sets the receive frame-synchronization error bit (RSYNCERR) in SPCR1. RSYNCERR can be cleared only by a receiver reset or by a write of 0 to this bit.

  If you want the McBSP to notify the CPU of receive frame-synchronization errors, you can set a special receive interrupt mode with the RINTM bits of SPCR1. When RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU each time that RSYNCERR is set.

15.5.3.2 Example of Unexpected Receive Frame-Synchronization Pulse

Figure 15-24 shows an unexpected receive frame-synchronization pulse during normal operation of the serial port, with time intervals between data packets. When the unexpected frame-synchronization pulse occurs, the RSYNCERR bit is set, the reception of data B is aborted, and the reception of data C begins. In addition, if RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU.

Figure 15-24. An Unexpected Frame-Synchronization Pulse During a McBSP Reception

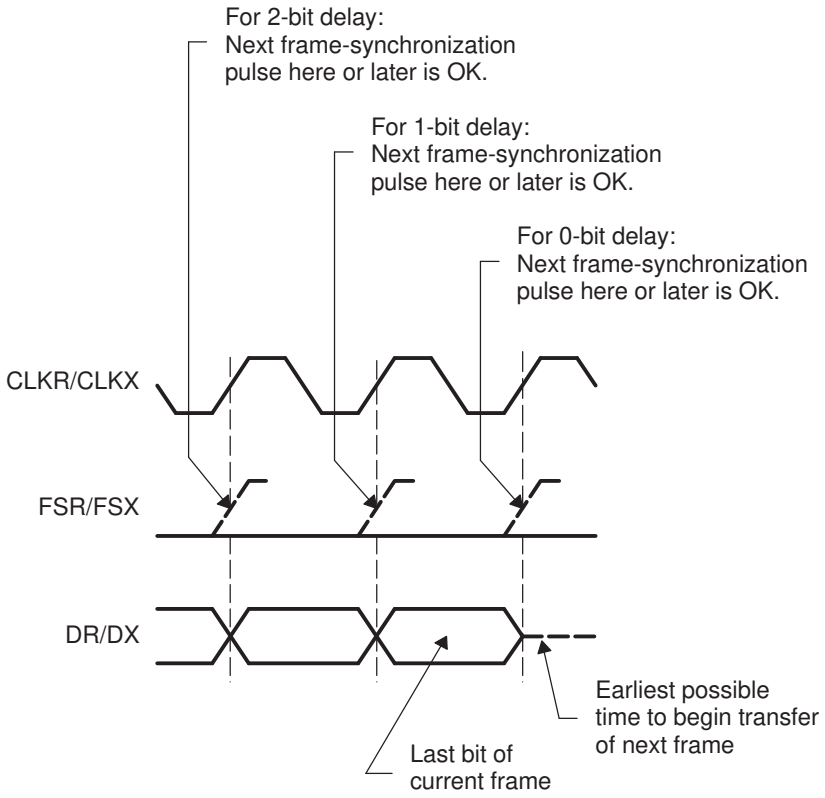


15.5.3.3 Preventing Unexpected Receive Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 MCLKR cycles, depending on the value in the RDATDLY bits of RCR2. For each possible data delay, Figure 15-25 shows when a new frame-synchronization pulse on FSR can safely occur relative to the last bit of the current frame.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-25. Proper Positioning of Frame-Synchronization Pulses



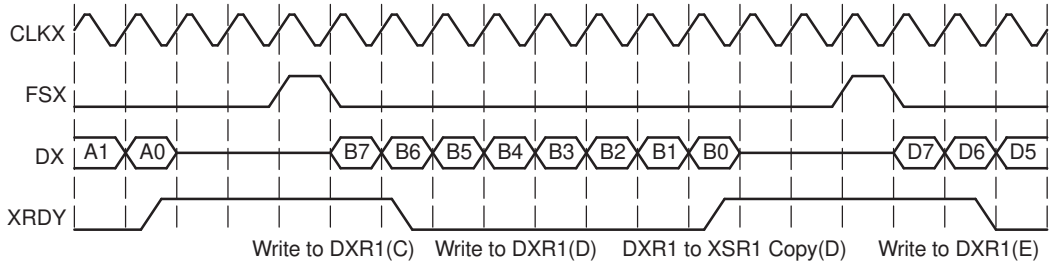
15.5.4 Overwrite in the Transmitter

As described in the section on McBSP transmission (page Section 15.3.6), the transmitter must copy the data previously written to the DXR(s) by the CPU or DMA controller into the XSR(s) and then shift each bit from the XSR(s) to the DX pin. If new data is written to the DXR(s) before the previous data is copied to the XSR(s), the previous data in the DXR(s) is overwritten and thus lost.

15.5.4.1 Example of Overwrite Condition

Figure 15-26 shows what happens if the data in DXR1 is overwritten before being transmitted. Initially, DXR1 is loaded with data C. A subsequent write to DXR1 overwrites C with D before C is copied to XSR1. Thus, C is never transmitted on DX.

Figure 15-26. Data in the McBSP Transmitter Overwritten and Thus Not Transmitted



15.5.4.2 Preventing Overwrites

You can prevent CPU overwrites by making the CPU:

- Poll for XRDY = 1 in SPCR2 before writing to the DXR(s). XRDY is set when data is copied from DXR1 to XSR1 and is cleared when new data is written to DXR1.

Copyright © 2012–2019, Texas Instruments Incorporated

- Wait for a transmit interrupt (XINT) before writing to the DXR(s). When XINTM = 00b in SPCR2, the transmitter sends XINT to the CPU each time XRDY is set.

You can prevent DMA overwrites by synchronizing DMA transfers to the transmit synchronization event XEVT. The transmitter sends an XEVT signal each time XRDY is set.

15.5.4.3 Underflow in the Transmitter

The McBSP indicates a transmitter empty (or underflow) condition by clearing the $\overline{\text{XEMPTY}}$ bit in SPCR2. Either of the following events activates $\overline{\text{XEMPTY}}$ ($\overline{\text{XEMPTY}}$ = 0):

- DXR1 has not been loaded since the last DXR-to-XSR copy, and all bits of the data word in the XSR(s) have been shifted out on the DX pin.
- The transmitter is reset (by forcing XRST = 0 in SPCR2, or by a device reset) and is then restarted.

In the underflow condition, the transmitter continues to transmit the old data that is in the DXR(s) for every new transmit frame-synchronization signal until a new value is loaded into DXR1 by the CPU or the DMA controller.

NOTE: If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs). If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

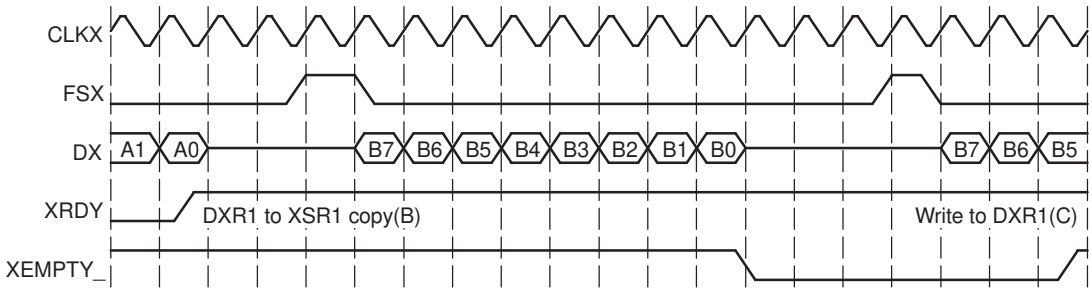
XEMPTY is deactivated (XEMPTY = 1) when a new word in DXR1 is transferred to XSR1. If FSXM = 1 in PCR and FSGM = 0 in SRGR2, the transmitter generates a single internal FSX pulse in response to a DXR-to-XSR copy. Otherwise, the transmitter waits for the next frame-synchronization pulse before sending out the next frame on DX.

When the transmitter is taken out of reset (XRST = 1), it is in a transmitter ready (XRDY = 1 in SPCR2) and transmitter empty (XEMPTY = 0) state. If DXR1 is loaded by the CPU or the DMA controller before internal FSX goes active high, a valid DXR-to-XSR transfer occurs. This allows for the first word of the first frame to be valid even before the transmit frame-synchronization pulse is generated or detected. Alternatively, if a transmit frame-synchronization pulse is detected before DXR1 is loaded, zeros are output on DX.

15.5.4.3.1 Example of the Underflow Condition

Figure 15-27 shows an underflow condition. After B is transmitted, DXR1 is not reloaded before the subsequent frame-synchronization pulse. Thus, B is again transmitted on DX.

Figure 15-27. Underflow During McBSP Transmission

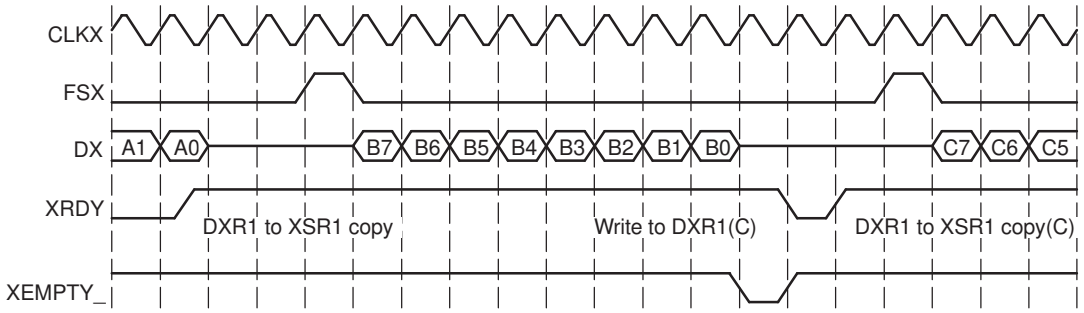


15.5.4.3.2 Example of Preventing Underflow Condition

Figure 15-28 shows the case of writing to DXR1 just before an underflow condition would otherwise occur. After B is transmitted, C is written to DXR1 before the next frame-synchronization pulse. As a result, there is no underflow; B is not transmitted twice.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-28. Underflow Prevented in the McBSP Transmitter



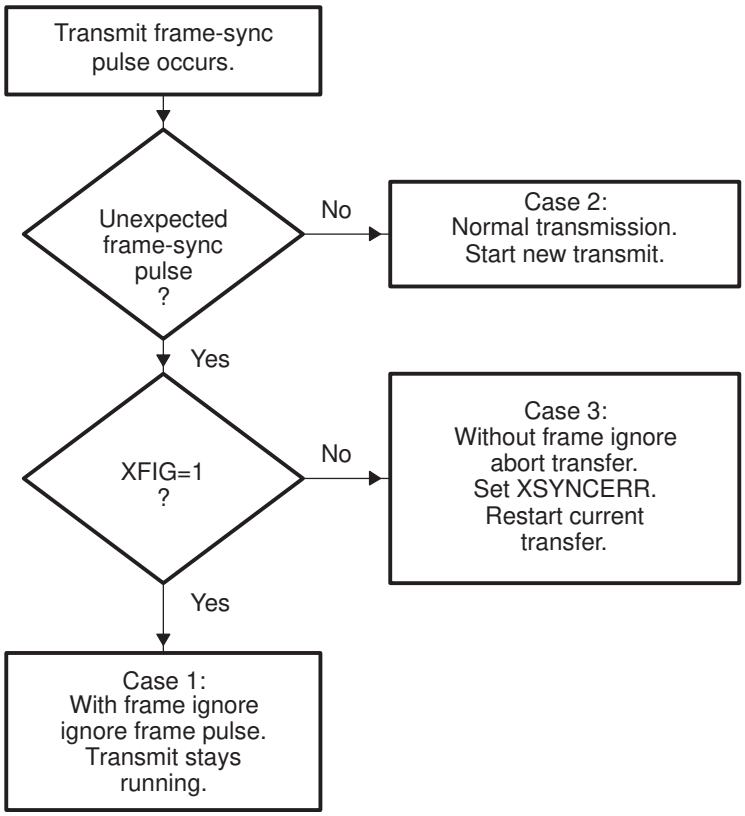
15.5.5 Unexpected Transmit Frame-Synchronization Pulse

Section 15.5.5.1 shows how the McBSP responds to any transmit frame-synchronization pulses, including an unexpected pulse. Section 15.5.5.2 and Section 15.5.5.3 show examples of a frame-synchronization error and an example of how to prevent such an error, respectively.

15.5.5.1 Possible Responses to Transmit Frame-Synchronization Pulses

Figure 15-29 shows the decision tree that the transmitter uses to handle all incoming frame-synchronization pulses. The figure assumes that the transmitter has been started (XRST = 1 in SPCR2). Case 3 in the figure is the case in which an error occurs.

Figure 15-29. Possible Responses to Transmit Frame-Synchronization Pulses



Any one of three cases can occur:

- Case 1: Unexpected internal FSX pulses with XFIG = 1 in XCR2. Transmit frame-synchronization pulses are ignored, and the transmission continues.
- Case 2: Normal serial port transmission. Transmission continues normally because the frame-

Copyright © 2012–2019, Texas Instruments Incorporated

synchronization pulse is not unexpected. There are two possible reasons why a transmit operations might *not* be in progress when the pulse occurs:

This FSX pulse is the first after the transmitter is enabled (XRST = 1).

The serial port is in the interpacket intervals. The programmed data delay for transmission (programmed with the XDATDLY bits of XCR2) may start during these interpacket intervals before the first bit of the previous word is transmitted. Thus, at maximum packet frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.

- Case 3: Unexpected transmit frame synchronization with XFIG = 0 (frame-synchronization pulses not ignored). Unexpected frame-synchronization pulses can originate from an external source or from the internal sample rate generator.

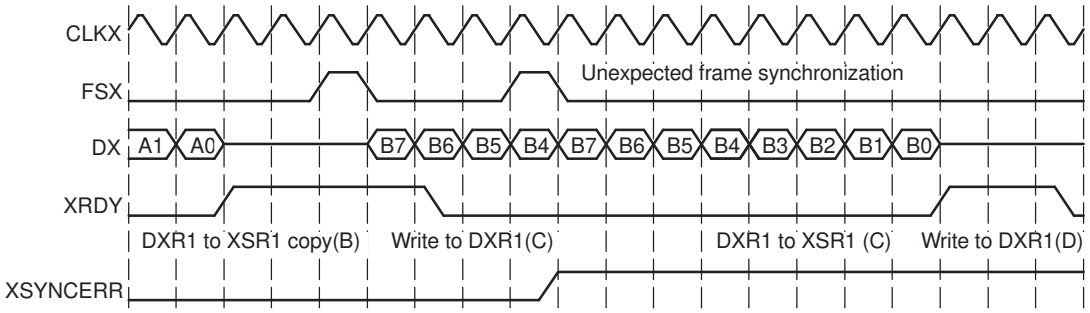
  If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse, and the transmitter sets the transmit frame-synchronization error bit (XSYNCERR) in SPCR2. XSYNCERR can be cleared only by a transmitter reset or by a write of 0 to this bit.

  If you want the McBSP to notify the CPU of frame-synchronization errors, you can set a special transmit interrupt mode with the XINTM bits of SPCR2. When XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU each time that XSYNCERR is set.

15.5.5.2 Example of Unexpected Transmit Frame-Synchronization Pulse

Figure 15-30 shows an unexpected transmit frame-synchronization pulse during normal operation of the serial port with intervals between the data packets. When the unexpected frame-synchronization pulse occurs, the XSYNCERR bit is set and the transmission of data B is restarted because no new data has been passed to XSR1 yet. In addition, if XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU.

Figure 15-30. An Unexpected Frame-Synchronization Pulse During a McBSP Transmission

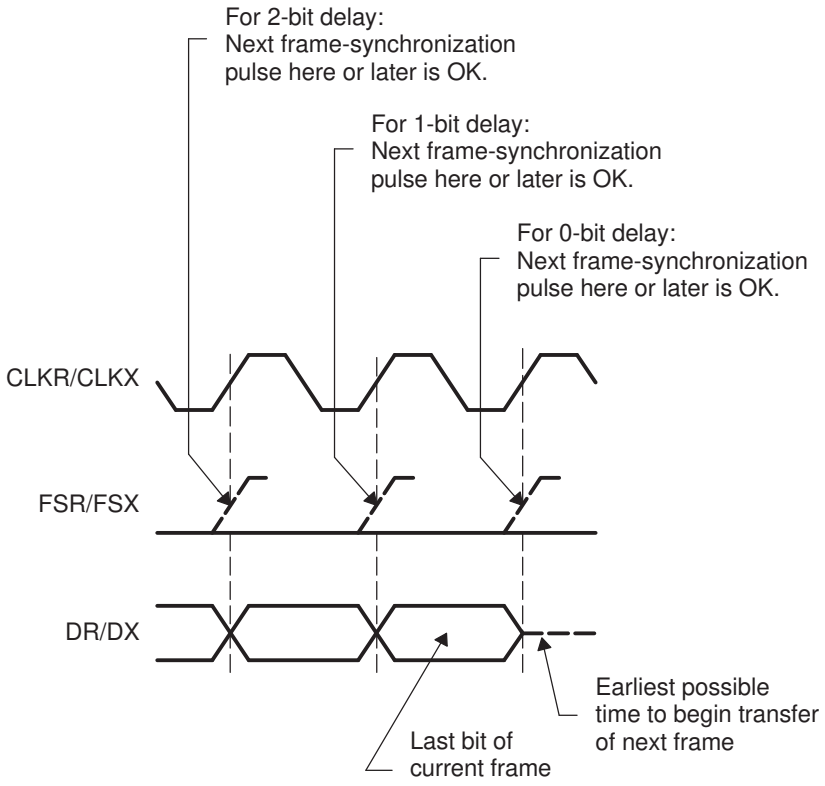


15.5.5.3 Preventing Unexpected Transmit Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKX cycles, depending on the value in the XDATDLY bits of XCR2. For each possible data delay, Figure 15-31 shows when a new frame-synchronization pulse on FSX can safely occur relative to the last bit of the current frame.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-31. Proper Positioning of Frame-Synchronization Pulses



15.6 Multichannel Selection Modes

This section discusses the multichannel selection modes for the McBSP.

15.6.1 Channels, Blocks, and Partitions

A McBSP channel is a time slot for shifting in/out the bits of one serial word. Each McBSP supports up to 128 channels for reception and 128 channels for transmission.

In the receiver and in the transmitter, the 128 available channels are divided into eight blocks that each contain 16 contiguous channels (see Table 15-8 through Table 15-10) :

- It is possible to have two receive partitions (A & B) and 8 transmit partitions (A – H).
- McBSP can transmit/receive on selected channels.
- Each channel partition has a dedicated channel-enable register. Each bit controls whether data flow is allowed or prevented in one of the channels assigned to that partition.
- There are three transmit multichannel modes and one receive multichannel mode.

Table 15-8. Block - Channel Assignment

Block	Channels
0	0 -15
1	16 - 31
2	32 - 47
3	48 - 63
4	64 - 79
5	80 - 95
6	96 - 111
7	112 - 127

Copyright © 2012–2019, Texas Instruments Incorporated

The blocks are assigned to partitions according to the selected partition mode. In the two-partition mode (described in Section 15.6.4), you assign one even-numbered block (0, 2, 4, or 6) to partition A and one odd-numbered block (1, 3, 5, or 7) to partition B. In the 8-partition mode (described in Section 15.6.5), blocks 0 through 7 are automatically assigned to partitions, A through H, respectively.

Table 15-9. 2-Partition Mode

Partition	Blocks
A	0 or 2 or 4 or 6
B	1 or 3 or 5 or 7

Table 15-10. 8-Partition mode

Partition	Blocks	Channels
A	0	0 -15
B	1	16 - 31
C	2	32 - 47
D	3	48 - 63
E	4	64 - 79
F	5	80 - 95
G	6	96 - 111
H	7	112 - 127

The number of partitions for reception and the number of partitions for transmission are independent. For example, it is possible to use two receive partitions (A and B) and eight transmit partitions (A-H).

15.6.2 Multichannel Selection

When a McBSP uses a time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices, the McBSP may need to receive and/or transmit on only a few channels. To save memory and bus bandwidth, you can use a multichannel selection mode to prevent data flow in some of the channels.

Each channel partition has a dedicated channel enable register. If the appropriate multichannel selection mode is on, each bit in the register controls whether data flow is allowed or prevented in one of the channels that is assigned to that partition.

The McBSP has one receive multichannel selection mode (described in Section 15.6.6) and three transmit multichannel selection modes (described in Section 15.6.7).

15.6.3 Configuring a Frame for Multichannel Selection

Before you enable a multichannel selection mode, make sure you properly configure the data frame:

- Select a single-phase frame (RPHASE/XPHASE = 0). Each frame represents a TDM data stream.
- Set a frame length (in RFRLEN1/XFRLEN1) that includes the highest-numbered channel to be used. For example, if you plan to use channels 0, 15, and 39 for reception, the receive frame length must be at least 40 (RFRLEN1 = 39). If XFRLEN1 = 39 in this case, the receiver creates 40 time slots per frame but only receives data during time slots 0, 15, and 39 of each frame.

15.6.4 Using Two Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions (described in Section 15.6.5). If you choose the 2-partition mode (RMCME = 0 for reception, XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame-synchronization pulse, the receiver or transmitter begins with the channels in partition A and then alternates between partitions B and A until the complete frame has been transferred. When the next frame-synchronization pulse occurs, the next frame is transferred beginning with the channels in partition A.

Copyright © 2012–2019, Texas Instruments Incorporated

15.6.4.1 Assigning Blocks to Partitions A and B

For reception, any two of the eight receive-channel blocks can be assigned to receive partitions A and B, which means up to 32 receive channels can be enabled at any given point in time. Similarly, any two of the eight transmit-channel blocks (up 32 enabled transmit channels) can be assigned to transmit partitions A and B.

For reception:

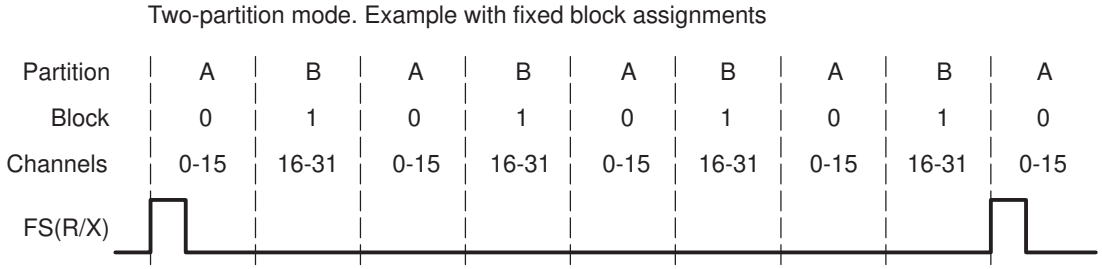
- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bits. In the receive multichannel selection mode (described in Section 15.6.6), the channels in this partition are controlled by receive channel enable register A (RCERA).
- Assign an odd-numbered block (1, 3, 5, or 7) to receive partition B with the RPBBLK bits. In the receive multichannel selection mode, the channels in this partition are controlled by receive channel enable register B (RCERB).

For transmission:

- Assign an even-numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the XPABLK bits. In one of the transmit multichannel selection modes (described in Section 15.6.7), the channels in this partition are controlled by transmit channel enable register A (XCERA).
- Assign an odd-numbered block (1, 3, 5, or 7) to transmit partition B with the XPBBLK bits. In one of the transmit multichannel selection modes, the channels in this partition are controlled by transmit channel enable register B (XCERB).

Figure 15-32 shows an example of alternating between the channels of partition A and the channels of partition B. Channels 0-15 have been assigned to partition A, and channels 16-31 have been assigned to partition B. In response to a frame-synchronization pulse, the McBSP begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.

Figure 15-32. Alternating Between the Channels of Partition A and the Channels of Partition B



As explained in Section 15.6.4.2, you can dynamically change which blocks of channels are assigned to the partitions.

15.6.4.2 Reassigning Blocks During Reception/Transmission

If you want to use more than 32 channels, you can change which channel blocks are assigned to partitions A and B during the course of a data transfer. However, these changes must be carefully timed. While a partition is being transferred, its associated block assignment bits cannot be modified and its associated channel enable register cannot be modified. For example, if block 3 is being transferred and block 3 is assigned to partition A, you can modify neither (R/X)PABLK to assign different channels to partition A nor (R/X)CERA to change the channel configuration for partition A.

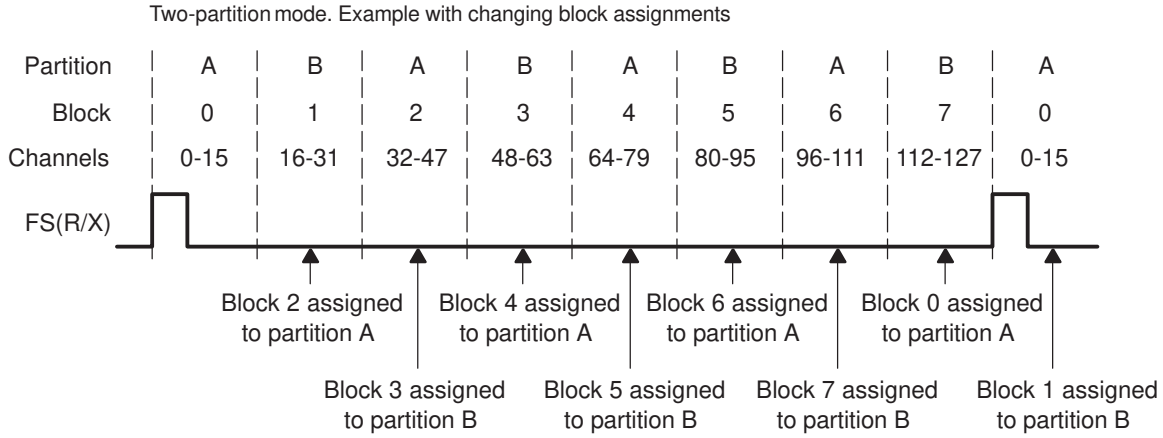
Copyright © 2012–2019, Texas Instruments Incorporated

Several features of the McBSP help you time the reassignment:

- The block of channels currently involved in reception/transmission (the current block) is reflected in the RCBLK/XCBLK bits. Your program can poll these bits to determine which partition is active. When a partition is not active, it is safe to change its block assignment and channel configuration.
- At the end of every block (at the boundary of two partitions), an interrupt can be sent to the CPU. In response to the interrupt, the CPU can then check the RCBLK/XCBLK bits and update the inactive partition. See Section 15.6.7.3, *Using Interrupts Between Block Transfers*.

Figure 15-33 shows an example of reassigning channels throughout a data transfer. In response to a frame-synchronization pulse, the McBSP alternates between partitions A and B. Whenever partition B is active, the CPU changes the block assignment for partition A. Whenever partition A is active, the CPU changes the block assignment for partition B.

Figure 15-33. Reassigning Channel Blocks Throughout a McBSP Data Transfer



15.6.5 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions (described in Section 15.6.4). If you choose the 8-partition mode (RMCME = 1 for reception, XMCME = 1 for transmission), McBSP channels are activated in the following order: A, B, C, D, E, F, G, H. In response to a frame-synchronization pulse, the receiver or transmitter begins with the channels in partition A and then continues with the other partitions in order until the complete frame has been transferred. When the next frame-synchronization pulse occurs, the next frame is transferred, beginning with the channels in partition A.

In the 8-partition mode, the (R/X)PABLK and (R/X)PBBLK bits are ignored and the 16-channel blocks are assigned to the partitions as shown in Table 15-11 and Table 15-12. These assignments cannot be changed. The tables also show the registers used to control the channels in the partitions.

Table 15-11. Receive Channel Assignment and Control With Eight Receive Partitions

Receive Partition	Assigned Block of Receive Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	RCERA
B	Block 1: channels 16 through 31	RCERB
C	Block 2: channels 32 through 47	RCERC
D	Block 3: channels 48 through 63	RCERD
E	Block 4: channels 64 through 79	RCERE
F	Block 5: channels 80 through 95	RCERF
G	Block 6: channels 96 through 111	RCERG
H	Block 7: channels 112 through 127	RCERH

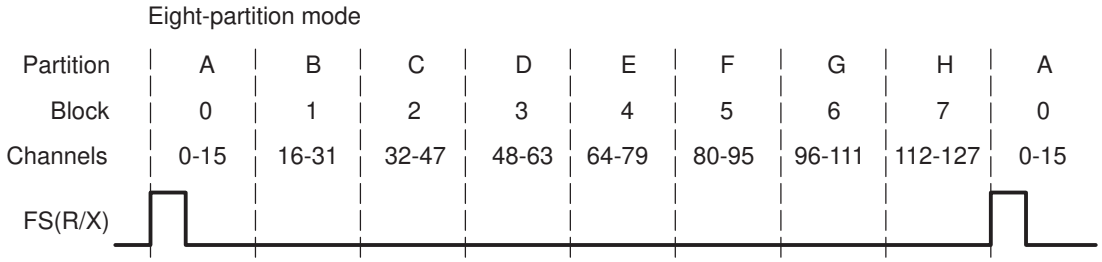
Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-12. Transmit Channel Assignment and Control When Eight Transmit Partitions Are Used

Transmit Partition	Assigned Block of Transmit Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	XCERA
B	Block 1: channels 16 through 31	XCERB
C	Block 2: channels 32 through 47	XCERC
D	Block 3: channels 48 through 63	XCERD
E	Block 4: channels 64 through 79	XCERE
F	Block 5: channels 80 through 95	XCERF
G	Block 6: channels 96 through 111	XCERG
H	Block 7: channels 112 through 127	XCERH

Figure 15-34 shows an example of the McBSP using the 8-partition mode. In response to a frame-synchronization pulse, the McBSP begins a frame transfer with partition A and then activates B, C, D, E, F, G, and H to complete a 128-word frame.

Figure 15-34. McBSP Data Transfer in the 8-Partition Mode



15.6.6 Receive Multichannel Selection Mode

The RMCM bit of MCR1 determines whether all channels or only selected channels are enabled for reception. When RMCM = 0, all 128 receive channels are enabled and cannot be disabled. When RMCM = 1, the receive multichannel selection mode is enabled. In this mode:

- Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit of MCR1.
- If a receive channel is disabled, any bits received in that channel are passed only as far as the receive buffer register(s) (RBR(s)). The receiver does not copy the content of the RBR(s) to the DRR(s), and as a result, does not set the receiver ready bit (RRDY). Therefore, no DMA synchronization event (REVT) is generated and, if the receiver interrupt mode depends on RRDY (RINTM = 00b), no interrupt is generated.

As an example of how the McBSP behaves in the receive multichannel selection mode, suppose you enable only channels 0, 15, and 39 and that the frame length is 40. The McBSP:

1. Accepts bits shifted in from the DR pin in channel 0
2. Ignores bits received in channels 1-14
3. Accepts bits shifted in from the DR pin in channel 15
4. Ignores bits received in channels 16-38
5. Accepts bits shifted in from the DR pin in channel 39

15.6.7 Transmit Multichannel Selection Modes

The XMCM bits of XCR2 determine whether all channels or only selected channels are enabled and unmasked for transmission. More details on enabling and masking are in Section 15.6.7.1. The McBSP has three transmit multichannel selection modes (XMCM = 01b, XMCM = 10b, and XMCM = 11b), which are described in the following table.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-13. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits

XMCM	Transmit Multichannel Selection Mode
00b	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.
01b	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked. The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs.
10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs.
11b	This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.

As an example of how the McBSP behaves in a transmit multichannel selection mode, suppose that XMCM = 01b (all channels disabled unless individually enabled) and that you have enabled only channels 0, 15, and 39. Suppose also that the frame length is 40. The McBSP:…

1. Shifts data to the DX pin in channel 0
2. Places the DX pin in the high impedance state in channels 1-14
3. Shifts data to the DX pin in channel 15
4. Places the DX pin in the high impedance state in channels 16-38
5. Shifts data to the DX pin in channel 39

15.6.7.1 Disabling/Enabling Versus Masking/Unmasking

For transmission, a channel can be:

- Enabled and unmasked (transmission can begin and can be completed)
- Enabled but masked (transmission can begin but cannot be completed)
- Disabled (transmission cannot occur)

The following definitions explain the channel control options:

Enabled channel A channel that can begin transmission by passing data from the data transmit register(s) (DXR(s)) to the transmit shift registers (XSR(s)).

Masked channel A channel that cannot complete transmission. The DX pin is held in the high impedance state; data cannot be shifted out on the DX pin.

In systems where symmetric transmit and receive provides software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for reception because multiple receptions cannot cause serial bus contention.

Disabled channel A channel that is not enabled. A disabled channel is also masked.

Because no DXR-to-XSR copy occurs, the XRDY bit of SPCR2 is not set. Therefore, no DMA synchronization event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 00b in SPCR2), no interrupt is generated.

The XEMPTY bit of SPCR2 is not affected.

Unmasked channel A channel that is not masked. Data in the XSR(s) is shifted out on the DX pin.

15.6.7.2 Activity on McBSP Pins for Different Values of XMCM

Figure 15-35 shows the activity on the McBSP pins for the various XMCM values. In all cases, the transmit frame is configured as follows:

- XPHASE = 0: Single-phase frame (required for multichannel selection modes)
- XFRLEN1 = 0000011b: 4 words per frame

Copyright © 2012–2019, Texas Instruments Incorporated

- XWDLEN1 = 000b: 8 bits per word
- XMCME = 0: 2-partition mode (only partitions A and B used)

In the case where XMCM = 11b, transmission and reception are symmetric, which means the corresponding bits for the receiver (RPHASE, RFRLEN1, RWDLEN1, and RMCME) must have the same values as XPHASE, XFRLEN1, and XWDLEN1, respectively.

In the figure, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

15.6.7.3 Using Interrupts Between Block Transfers

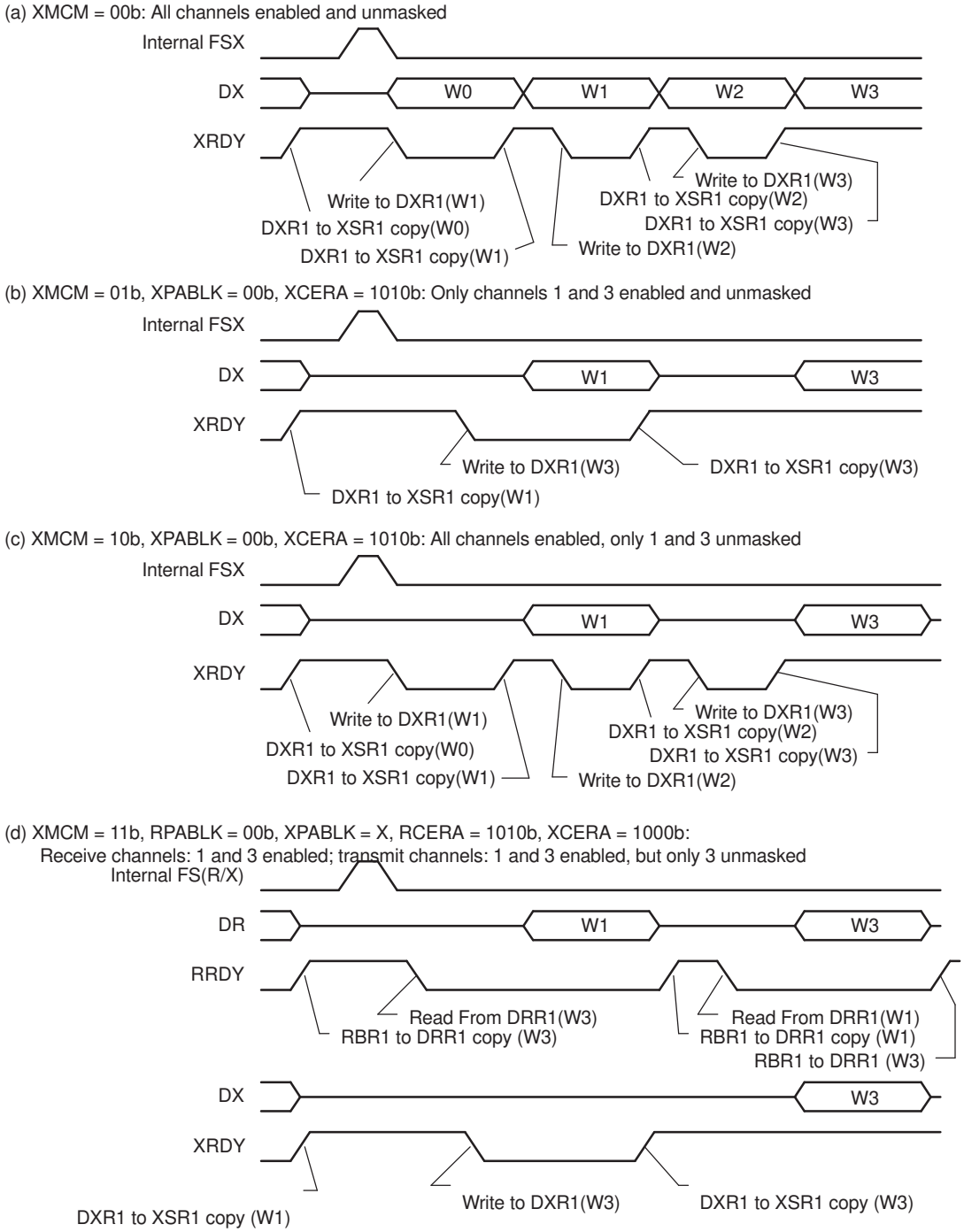
When a multichannel selection mode is used, an interrupt request can be sent to the CPU at the end of every 16-channel block (at the boundary between partitions and at the end of the frame). In the receive multichannel selection mode, a receive interrupt (RINT) request is generated at the end of each block transfer if RINTM = 01b. In any of the transmit multichannel selection modes, a transmit interrupt (XINT) request is generated at the end of each block transfer if XINTM = 01b. When RINTM/XINTM = 01b, no interrupt is generated unless a multichannel selection mode is on.

These interrupt pulses are active high and last for two CPU clock cycles.

This type of interrupt is especially helpful if you are using the two-partition mode (described in Section 15.6.4) and you want to know when you can assign a different block of channels to partition A or B.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-35. Activity on McBSP Pins for the Possible Values of XMCM



15.7 SPI Operation Using the Clock Stop Mode

This section explains how to use the McBSP in SPI mode.

15.7.1 SPI Protocol

The SPI protocol is a master-slave configuration with one master device and one or more slave devices. The interface consists of the following four signals:

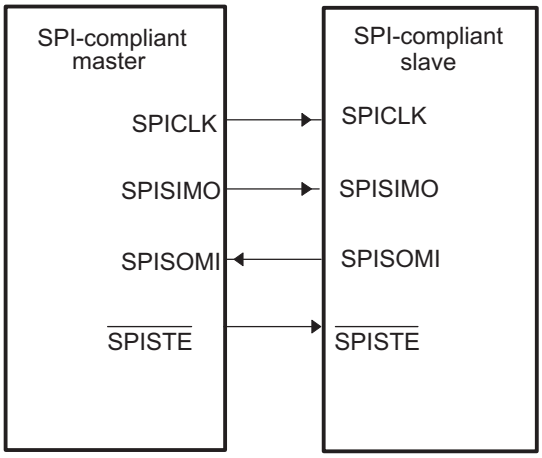
- Serial data input (also referred to as slave out/master in, or SOMI)

Copyright © 2012–2019, Texas Instruments Incorporated

- Serial data output (also referred to as slave in/master out, or SIMO)
- Shift-clock (also referred to as SPICLK)
- Slave-enable signal (also referred to as $\overline{\text{SPISTE}}$)

A typical SPI interface with a single slave device is shown in Figure 15-36.

Figure 15-36. Typical SPI Interface



The master device controls the flow of communication by providing shift-clock and slave-enable signals. The slave-enable signal is an optional active-low signal that enables the serial data input and output of the slave device (device not sending out the clock).

In the absence of a dedicated slave-enable signal, communication between the master and slave is determined by the presence or absence of an active shift-clock. When the McBSP is operating in SPI master mode and the $\overline{\text{SPISTE}}$ signal is not used by the slave SPI port, the slave device must remain enabled at all times, and multiple slaves cannot be used.

15.7.2 Clock Stop Mode

The clock stop mode of the McBSP provides compatibility with the SPI protocol. When the McBSP is configured in clock stop mode, the transmitter and receiver are internally synchronized so that the McBSP functions as an SPI master or slave device. The transmit clock signal (CLKX) corresponds to the serial clock signal (SPICLK) of the SPI protocol, while the transmit frame-synchronization signal (FSX) is used as the slave-enable signal ($\overline{\text{SPISTE}}$).

The receive clock signal (MCLKR) and receive frame-synchronization signal (FSR) are not used in the clock stop mode because these signals are internally connected to their transmit counterparts, CLKX and FSX.

15.7.3 Bits Used to Enable and Configure the Clock Stop Mode

The bits required to configure the McBSP as an SPI device are introduced in Table 15-14. Table 15-15 shows how the various combinations of the CLKSTP bit and the polarity bits CLKXP and CLKRP create four possible clock stop mode configurations. The timing diagrams in Section 15.7.4 show the effects of CLKSTP, CLKXP, and CLKRP.

Table 15-14. Bits Used to Enable and Configure the Clock Stop Mode

Bit Field	Description
CLKSTP bits of SPCR1	Use these bits to enable the clock stop mode and to select one of two timing variations. (See also Table 15-15.)
CLKXP bit of PCR	This bit determines the polarity of the CLKX signal. (See also Table 15-15.)
CLKRP bit of PCR	This bit determines the polarity of the MCLKR signal. (See also Table 15-15.)
CLKXM bit of PCR	This bit determines whether CLKX is an input signal (McBSP as slave) or an output signal (McBSP as master).

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-14. Bits Used to Enable and Configure the Clock Stop Mode (continued)

Bit Field	Description
XPHASE bit of XCR2	You must use a single-phase transmit frame (XPHASE = 0).
RPHASE bit of RCR2	You must use a single-phase receive frame (RPHASE = 0).
XFRLEN1 bits of XCR1	You must use a transmit frame length of 1 serial word (XFRLEN1 = 0).
RFRLEN1 bits of RCR1	You must use a receive frame length of 1 serial word (RFRLEN1 = 0).
XWDLEN1 bits of XCR1	The XWDLEN1 bits determine the transmit packet length. XWDLEN1 must be equal to RWDLEN1 because in the clock stop mode. The McBSP transmit and receive circuits are synchronized to a single clock.
RWDLEN1 bits of RCR1	The RWDLEN1 bits determine the receive packet length. RWDLEN1 must be equal to XWDLEN1 because in the clock stop mode. The McBSP transmit and receive circuits are synchronized to a single clock.

Table 15-15. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of MCLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of MCLKR.

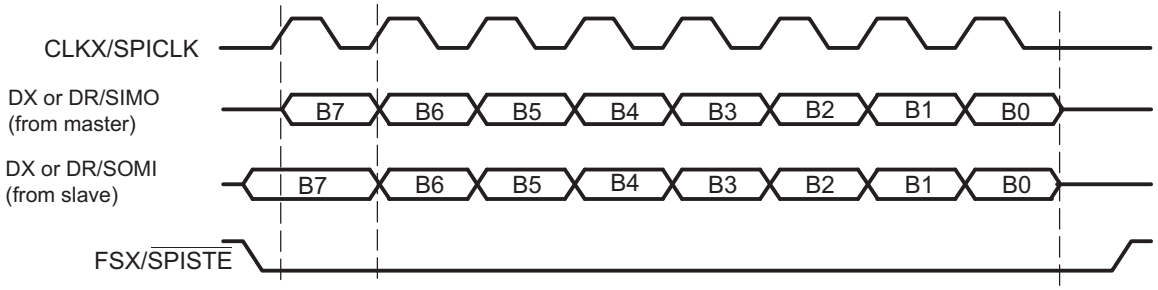
15.7.4 Clock Stop Mode Timing Diagrams

The timing diagrams for the four possible clock stop mode configurations are shown here. Notice that the frame-synchronization signal used in clock stop mode is active throughout the entire transmission as a slave-enable signal. Although the timing diagrams show 8-bit transfers, the packet length can be set to 8, 12, 16, 20, 24, or 32 bits per packet. The receive packet length is selected with the RWDLEN1 bits of RCR1, and the transmit packet length is selected with the XWDLEN1 bits of XCR1. For clock stop mode, the values of RWDLEN1 and XWDLEN1 must be the same because the McBSP transmit and receive circuits are synchronized to a single clock.

NOTE: Even if multiple words are consecutively transferred, the CLKX signal is always stopped and the FSX signal returns to the inactive state after a packet transfer. When consecutive packet transfers are performed, this leads to a minimum idle time of two bit-periods between each packet transfer.

Copyright © 2012–2019, Texas Instruments Incorporated

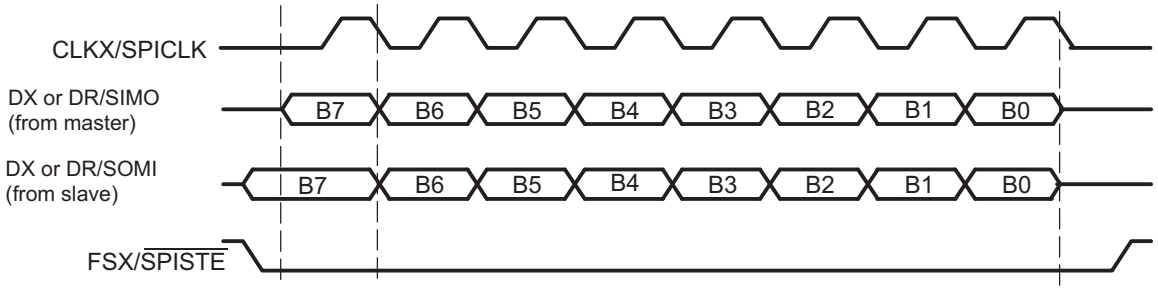
Figure 15-37. SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 0, and CLKRP = 0



A If the McBSP is the SPI master (CLKXM = 1), SIMO = DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.

B If the McBSP is the SPI master (CLKXM = 1), SOMI = DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

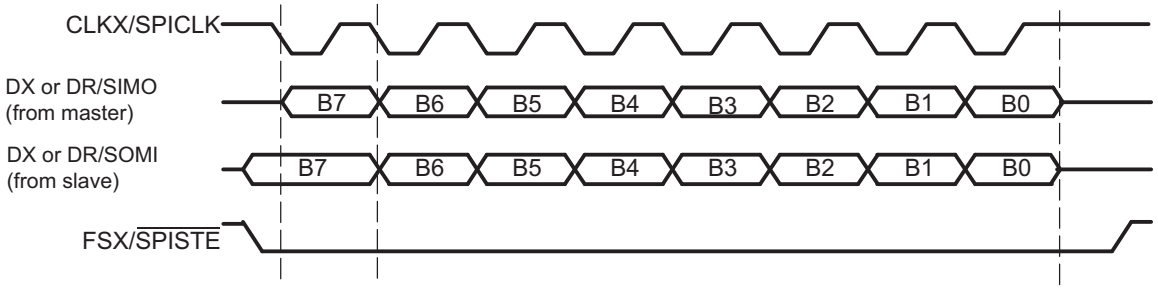
Figure 15-38. SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 0, CLKRP = 1



A If the McBSP is the SPI master (CLKXM = 1), SIMO = DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.

B If the McBSP is the SPI master (CLKXM = 1), SOMI = DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

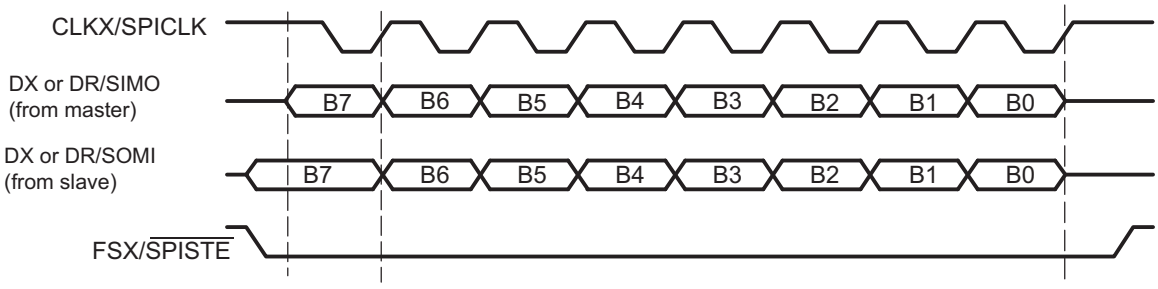
Figure 15-39. SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 1, and CLKRP = 0



A If the McBSP is the SPI master (CLKXM = 1), SIMO = DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.

B If the McBSP is the SPI master (CLKXM = 1), SOMI = DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

Figure 15-40. SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 1, CLKRP = 1



A If the McBSP is the SPI master (CLKXM = 1), SIMO=DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.

B If the McBSP is the SPI master (CLKXM = 1), SOMI=DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

Copyright © 2012–2019, Texas Instruments Incorporated

15.7.5 Procedure for Configuring a McBSP for SPI Operation

To configure the McBSP for SPI master or slave operation:

Step 1. Place the transmitter and receiver in reset.

Clear the transmitter reset bit (XRST = 0) in SPCR2 to reset the transmitter. Clear the receiver reset bit (RRST = 0) in SPCR1 to reset the receiver.

Step 2. Place the sample rate generator in reset.

Clear the sample rate generator reset bit (GRST = 0) in SPCR2 to reset the sample rate generator.

Step 3. Program registers that affect SPI operation.

Program the appropriate McBSP registers to configure the McBSP for proper operation as an SPI master or an SPI slave. For a list of important bits settings, see one of the following topics:

- *McBSP as the SPI Master* (Section 15.7.6)
- *McBSP as an SPI Slave* (Section 15.7.7)

Step 4. Enable the sample rate generator.

To release the sample rate generator from reset, set the sample rate generator reset bit (GRST = 1) in SPCR2.

Make sure that during the write to SPCR2, you only modify GRST. Otherwise, you modify the McBSP configuration you selected in the previous step.

Step 5. Enable the transmitter and receiver.

After the sample rate generator is released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.

If the CPU services the McBSP transmit and receive buffers, then you can immediately enable the transmitter (XRST = 1 in SPCR2) and enable the receiver (RRST = 1 in SPCR1).

If the DMA controller services the McBSP transmit and receive buffers, then you must first configure the DMA controller (this includes enabling the channels that service the McBSP buffers). When the DMA controller is ready, make XRST = 1 and RRST = 1.

In either case, make sure you only change XRST and RRST when you write to SPCR2 and SPCR1. Otherwise, you modify the bit settings you selected earlier in this procedure.

After the transmitter and receiver are released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.

Step 6. If necessary, enable the frame-synchronization logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1,2] is loaded with data), set FRST = 1 if an internally generated frame-synchronization pulse is required (that is, if the McBSP is the SPI master).

15.7.6 McBSP as the SPI Master

An SPI interface with the McBSP used as the master is shown in Figure 15-41. When the McBSP is configured as a master, the transmit output signal (DX) is used as the SIMO signal of the SPI protocol and the receive input signal (DR) is used as the SOMI signal.

The register bit values required to configure the McBSP as a master are listed in Table 15-16. After the table are more details about the configuration requirements.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-41. SPI Interface with McBSP Used as Master

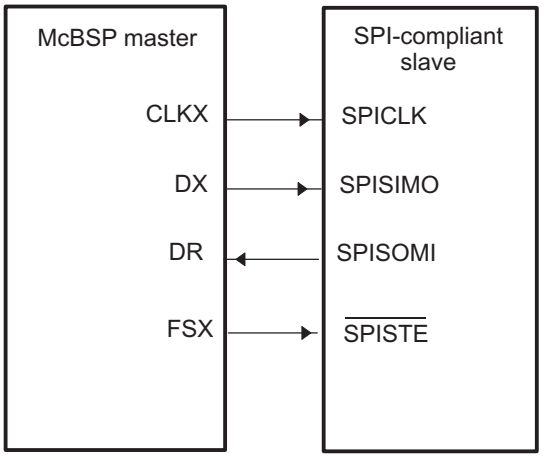


Table 15-16. Bit Values Required to Configure the McBSP as a SPI Master

Required Bit Setting	Description
CLKSTP = 10b or 11b	The clock stop mode (without or with a clock delay) is selected.
CLKXP = 0 or 1	The polarity of CLKX as seen on the MCLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of MCLKR as seen on the MCLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 1	The MCLKX pin is an output pin driven by the internal sample rate generator. Because CLKSTP is equal to 10b or 11b, MCLKR is driven internally by CLKX.
SCLKME = 0	The clock generated by the sample rate generator (CLKG) is derived from the CPU clock.
CLKSM = 1	
CLKGDV is a value from 1 to 255	CLKGDV defines the divide down value for CLKG.
FSXM = 1	The FSX pin is an output pin driven according to the FSGM bit.
FSGM = 0	The transmitter drives a frame-synchronization pulse on the FSX pin every time data is transferred from DXR1 to XSR1.
FSXP = 1	The FSX pin is active low.
XDATDLY = 01b	This setting provides the correct setup time on the FSX signal.
RDATDLY = 01b	

When the McBSP functions as the SPI master, it controls the transmission of data by producing the serial clock signal. The clock signal on the MCLKX pin is enabled only during packet transfers. When packets are not being transferred, the MCLKX pin remains high or low depending on the polarity used.

For SPI master operation, the MCLKX pin must be configured as an output. The sample rate generator is then used to derive the CLKX signal from the CPU clock. The clock stop mode internally connects the MCLKX pin to the MCLKR signal so that no external signal connection is required on the MCLKR pin and both the transmit and receive circuits are clocked by the master clock (CLKX).

The data delay parameters of the McBSP (XDATDLY and RDATDLY) must be set to 1 for proper SPI master operation. A data delay value of 0 or 2 is undefined in the clock stop mode.

The McBSP can also provide a slave-enable signal ($\overline{\text{SPISTE}}$) on the FSX pin. If a slave-enable signal is required, the FSX pin must be configured as an output and the transmitter must be configured so that a frame-synchronization pulse is generated automatically each time a packet is transmitted (FSGM = 0). The polarity of the FSX pin is programmable high or low; however, in most cases the pin must be configured active low.

Copyright © 2012–2019, Texas Instruments Incorporated

When the McBSP is configured as described for SPI-master operation, the bit fields for frame-synchronization pulse width (FWID) and frame-synchronization period (FPER) are overridden, and custom frame-synchronization waveforms are not allowed. To see the resulting waveform produced on the FSX pin, see the timing diagrams in Section 15.7.4. The signal becomes active before the first bit of a packet transfer, and remains active until the last bit of the packet is transferred. After the packet transfer is complete, the FSX signal returns to the inactive state.

15.7.7 McBSP as an SPI Slave

An SPI interface with the McBSP used as a slave is shown in Figure 15-42. When the McBSP is configured as a slave, DX is used as the SOMI signal and DR is used as the SIMO signal.

The register bit values required to configure the McBSP as a slave are listed in Table 15-17. Following the table are more details about configuration requirements.

Figure 15-42. SPI Interface With McBSP Used as Slave

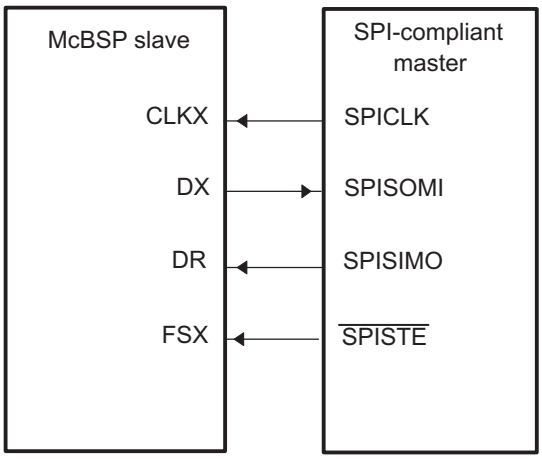


Table 15-17. Bit Values Required to Configure the McBSP as an SPI Slave

Required Bit Setting	Description
CLKSTP = 10b or 11b	The clock stop mode (without or with a clock delay) is selected.
CLKXP = 0 or 1	The polarity of CLKX as seen on the MCLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of MCLKR as seen on the MCLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 0	The MCLKX pin is an input pin, so that it can be driven by the SPI master. Because CLKSTP = 10b or 11b, MCLKR is driven internally by CLKX.
SCLKME = 0 CLKSM = 1	The clock generated by the sample rate generator (CLKG) is derived from the CPU clock. (The sample rate generator is used to synchronize the McBSP logic with the externally-generated master clock.)
CLKGDV = 1	The sample rate generator divides the CPU clock before generating CLKG.
FSXM = 0	The FSX pin is an input pin, so that it can be driven by the SPI master.
FSXP = 1	The FSX pin is active low.
XDATDLY = 00b	These bits must be 0s for SPI slave operation.
RDATDLY = 00b	

When the McBSP is used as an SPI slave, the master clock and slave-enable signals are generated externally by a master device. Accordingly, the CLKX and FSX pins must be configured as inputs. The MCLKX pin is internally connected to the MCLKR signal, so that both the transmit and receive circuits of the McBSP are clocked by the external master clock. The FSX pin is also internally connected to the FSR signal, and no external signal connections are required on the MCLKR and FSR pins.

Copyright © 2012–2019, Texas Instruments Incorporated

Although the CLKX signal is generated externally by the master and is asynchronous to the McBSP, the sample rate generator of the McBSP must be enabled for proper SPI slave operation. The sample rate generator must be programmed to its maximum rate of half the CPU clock rate. The internal sample rate clock is then used to synchronize the McBSP logic to the external master clock and slave-enable signals.

The McBSP requires an active edge of the slave-enable signal on the FSX input for each transfer. This means that the master device must assert the slave-enable signal at the beginning of each transfer, and deassert the signal after the completion of each packet transfer; the slave-enable signal cannot remain active between transfers. Unlike the standard SPI, this pin cannot be tied low all the time.

The data delay parameters of the McBSP must be set to 0 for proper SPI slave operation. A value of 1 or 2 is undefined in the clock stop mode.

15.8 Receiver Configuration

To configure the McBSP receiver, perform the following procedure:

1. Place the McBSP/receiver in reset (see Section 15.8.2).
2. Program McBSP registers for the desired receiver operation (see Section 15.8.1).
3. Take the receiver out of reset (see Section 15.8.2).

15.8.1 Programming the McBSP Registers for the Desired Receiver Operation

The following is a list of important tasks to be performed when you are configuring the McBSP receiver. Each task corresponds to one or more McBSP register bit fields.

- Global behavior:
  - Set the receiver pins to operate as McBSP pins.
  - Enable/disable the digital loopback mode.
  - Enable/disable the clock stop mode.
  - Enable/disable the receive multichannel selection mode.
- Data behavior:
  - Choose 1 or 2 phases for the receive frame.
  - Set the receive word length(s).
  - Set the receive frame length.
  - Enable/disable the receive frame-synchronization ignore function.
  - Set the receive companding mode.
  - Set the receive data delay.
  - Set the receive sign-extension and justification mode.
  - Set the receive interrupt mode.
- Frame-synchronization behavior:
  - Set the receive frame-synchronization mode.
  - Set the receive frame-synchronization polarity.
  - Set the sample rate generator (SRG) frame-synchronization period and pulse width.
- Clock behavior:
  - Set the receive clock mode.
  - Set the receive clock polarity.
  - Set the SRG clock divide-down value.
  - Set the SRG clock synchronization mode.
  - Set the SRG clock mode (choose an input clock).
  - Set the SRG input clock polarity.

Copyright © 2012–2019, Texas Instruments Incorporated

15.8.2 Resetting and Enabling the Receiver

The first step of the receiver configuration procedure is to reset the receiver, and the last step is to enable the receiver (to take it out of reset). Table 15-18 describes the bits used for both of these steps.

Table 15-18. Register Bits Used to Reset or Enable the McBSP Receiver Field Descriptions

Register	Bit	Field	Value	Description
SPCR2	7	FRST		Frame-synchronization logic reset
			0	Frame-synchronization logic is reset. The sample rate generator does not generate frame-synchronization signal FSG, even if GRST = 1.
			1	If GRST = 1, frame-synchronization signal FSG is generated after (FPER + 1) number of CLKG clock cycles; all frame counters are loaded with their programmed values.
SPCR2	6	GRST		Sample rate generator reset
			0	Sample rate generator is reset. If GRST = 0 due to a DSP reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 due to program code, CLKG and FSG are both driven low (inactive).
			1	Sample rate generator is enabled. CLKG is driven according to the configuration programmed in the sample rate generator registers (SRGR[1,2]). If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers.
SPCR1	0	RRST		Receiver reset
			0	The serial port receiver is disabled and in the reset state.
			1	The serial port receiver is enabled.

15.8.2.1 Reset Considerations

The serial port can be reset in the following two ways:

1. The DSP reset ($\overline{\text{XRS}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed ($\overline{\text{XRS}}$ signal released), GRST = FRST = RRST = XRST = 0 keep the entire serial port in the reset state, provided the McBSP clock is turned on.
2. The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the serial port control registers. The sample rate generator can be reset directly using the GRST bit in SPCR2.

Table 15-19 shows the state of McBSP pins when the serial port is reset due to a device reset and a direct receiver/transmitter reset.

For more details about McBSP reset conditions and effects, see Section 15.10.2, *Resetting and Initializing a McBSP*.

Table 15-19. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced By Device Reset	State Forced By Receiver Reset (RRST = 0 and GRST = 1)
MDRx	I	GPIO Input	Input
MCLKRx	I/O/Z	GPIO Input	Known state if input; MCLKR running if output
MFSRx	I/O/Z	GPIO Input	Known state if input; FSRP inactive state if output
			Transmitter reset (XRST = 0 and GRST = 1)
MDXx	O/Z	GPIO Input	Low impedance after transmit bit clock provided
MCLKXx	I/O/Z	GPIO Input	Known state if input; CLKX running if output
MFSXx	I/O/Z	GPIO Input	Known state if input; FSXP inactive state if output

15.8.3 Set the Receiver Pins to Operate as McBSP Pins

To configure a pin for its McBSP function , you should configure the bits of the GPxMUXn register appropriately. In addition to this, bits 12 and 13 of the PCR register must be set to 0. These bits are defined as reserved.

Copyright © 2012–2019, Texas Instruments Incorporated

15.8.4 Enable/Disable the Digital Loopback Mode

The DLB bit determines whether the digital loopback mode is on. DLB is described in Table 15-20.

Table 15-20. Register Bit Used to Enable/Disable the Digital Loopback Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	15	DLB	Digital loopback mode		R/W	0
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled.		

15.8.4.1 Digital Loopback Mode

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in Table 15-21. This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

Table 15-21. Receive Signals Connected to Transmit Signals in Digital Loopback Mode

This Receive Signal	Is Fed Internally by This Transmit Signal
MDR (receive data)	MDX (transmit data)
MFSR (receive frame synchronization)	MFSX (transmit frame synchronization)
MCLKR (receive clock)	MCLKX (transmit clock)

15.8.5 Enable/Disable the Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on. CLKSTP is described in Table 15-22.

Table 15-22. Register Bits Used to Enable/Disable the Clock Stop Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	12-11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; normal clocking for non-SPI mode		
			CLKSTP = 10b	Clock stop mode enabled, without clock delay		
			CLKSTP = 11b	Clock stop mode enabled, with clock delay		

15.8.5.1 Clock Stop Mode

The clock stop mode supports the SPI master-slave protocol. If you do not plan to use the SPI protocol, you can clear CLKSTP to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). The CLKXP bit determines whether the starting edge of the clock on the MCLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the MCLKR pin.

Table 15-23 summarizes the impact of CLKSTP, CLKXP, and CLKRP on serial port operation. In the clock stop mode, the receive clock is tied internally to the transmit clock, and the receive frame-synchronization signal is tied internally to the transmit frame-synchronization signal.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-23. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of MCLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of MCLKR.

15.8.6 Enable/Disable the Receive Multichannel Selection Mode

The RMCM bit determines whether the receive multichannel selection mode is on. RMCM is described in Table 15-24. For more details, see Section 15.6.6, *Receive Multichannel Selection Mode*.

Table 15-24. Register Bit Used to Enable/Disable the Receive Multichannel Selection Mode

Register	Bit	Name	Function		Type	Reset Value
MCR1	0	RMCM	Receive multichannel selection mode		R/W	0
			RMCM = 0	The mode is disabled. All 128 channels are enabled.		
			RMCM = 1	The mode is enabled. Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit.		

15.8.7 Choose One or Two Phases for the Receive Frame

The RPHASE bit (see Table 15-25) determines whether the receive data frame has one or two phases.

Table 15-25. Register Bit Used to Choose One or Two Phases for the Receive Frame

Register	Bit	Name	Function		Type	Reset Value
RCR2	15	RPHASE	Receive phase number		R/W	0
			Specifies whether the receive frame has 1 or 2 phases.			
			RPHASE = 0	Single-phase frame		
			RPHASE = 1	Dual-phase frame		

Copyright © 2012–2019, Texas Instruments Incorporated

15.8.8 Set the Receive Word Length(s)

The RWDLEN1 and RWDLEN2 bit fields (see Table 15-26) determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the receive data frame.

Table 15-26. Register Bits Used to Set the Receive Word Length(s)

Register	Bit	Name	Function		Type	Reset Value
RCR1	7-5	RWDLEN1	Receive word length 1		R/W	000
			Specifies the length of every serial word in phase 1 of the receive frame.			
			RWDLEN1 = 000	8 bits		
			RWDLEN1 = 001	12 bits		
			RWDLEN1 = 010	16 bits		
			RWDLEN1 = 011	20 bits		
			RWDLEN1 = 100	24 bits		
			RWDLEN1 = 101	32 bits		
			RWDLEN1 = 11X	Reserved		
RCR2	7-5	RWDLEN2	Receive word length 2		R/W	000
			If a dual-phase frame is selected, RWDLEN2 specifies the length of every serial word in phase 2 of the frame.			
			RWDLEN2 = 000	8 bits		
			RWDLEN2 = 001	12 bits		
			RWDLEN2 = 010	16 bits		
			RWDLEN2 = 011	20 bits		
			RWDLEN2 = 100	24 bits		
			RWDLEN2 = 101	32 bits		
			RWDLEN2 = 11X	Reserved		

15.8.8.1 Word Length Bits

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame and RWDLEN2 determines the word length in phase 2 of the frame.

15.8.9 Set the Receive Frame Length

The RFRLEN1 and RFRLEN2 bit fields (see Table 15-27) determine how many serial words are in phase 1 and in phase 2, respectively, of the receive data frame.

Table 15-27. Register Bits Used to Set the Receive Frame Length

Register	Bit	Name	Function		Type	Reset Value
RCR1	14-8	RFRLEN1	Receive frame length 1		R/W	000 0000
			(RFRLEN1 + 1) is the number of serial words in phase 1 of the receive frame.			
			RFRLEN1 = 000 0000	1 word in phase 1		
			RFRLEN1 = 000 0001	2 words in phase 1		
			|	|		
			|	|		
			RFRLEN1 = 111 1111	128 words in phase 1		

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-27. Register Bits Used to Set the Receive Frame Length (continued)

Register	Bit	Name	Function		Type	Reset Value
RCR2	14-8	RFRLEN2	Receive frame length 2		R/W	000 0000
			If a dual-phase frame is selected, (RFRLEN2 + 1) is the number of serial words in phase 2 of the receive frame.			
			RFRLEN2 = 000 0000	1 word in phase 2		
			RFRLEN2 = 000 0001	2 words in phase 2		
			|	|		
			|	|		
			RFRLEN2 = 111 1111	128 words in phase 2		

15.8.9.1 Selected Frame Length

The receive frame length is the number of serial words in the receive frame. Each frame can have one or two phases, depending on value that you load into the RPHASE bit.

If a single-phase frame is selected (RPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (RPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 15-28 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Program the RFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For the example, if you want a phase length of 128 words in phase 1, load 127 into RFRLEN1.

Table 15-28. How to Calculate the Length of the Receive Frame

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 ≤ RFRLEN1 ≤ 127	Don't care	(RFRLEN1 + 1) words
1	0 ≤ RFRLEN1 ≤ 127	0 ≤ RFRLEN2 ≤ 127	(RFRLEN1 + 1) + (RFRLEN2 + 1) words

15.8.10 Enable/Disable the Receive Frame-Synchronization Ignore Function

The RFIG bit (see Table 15-29) controls the receive frame-synchronization ignore function.

Table 15-29. Register Bit Used to Enable/Disable the Receive Frame-Synchronization Ignore Function

Register	Bit	Name	Function		Type	Reset Value
RCR2	2	RFIG	Receive frame-synchronization ignore		R/W	0
			RFIG = 0	An unexpected receive frame-synchronization pulse causes the McBSP to restart the frame transfer.		
			RFIG = 1	The McBSP ignores unexpected receive frame-synchronization pulses.		

15.8.10.1 Unexpected Frame-Synchronization Pulses and the Frame-Synchronization Ignore Function

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse.

When RFIG = 1, reception continues, ignoring the unexpected frame-synchronization pulses.

Copyright © 2012–2019, Texas Instruments Incorporated

When RFIG = 0, an unexpected FSR pulse causes the McBSP to discard the contents of RSR[1,2] in favor of the new incoming data. Therefore, if RFIG = 0 and an unexpected frame-synchronization pulse occurs, the serial port:

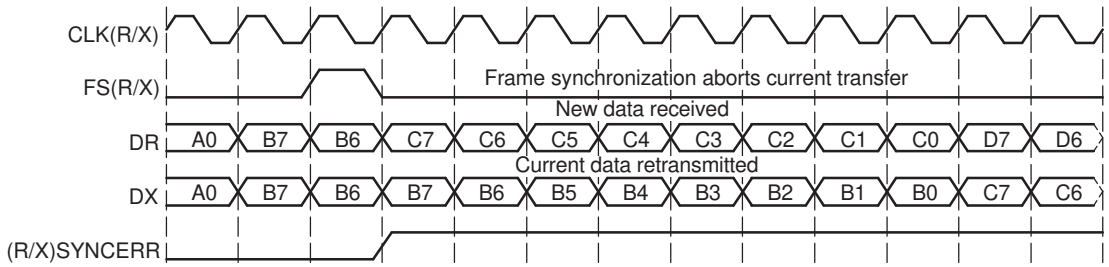
1. Aborts the current data transfer
2. Sets RSYNCERR in SPCR1 to 1
3. Begins the transfer of a new data word

For more details about the frame-synchronization error condition, see Section 15.5.3, *Unexpected Receive Frame-Synchronization Pulse*.

15.8.10.2 Examples of Effects of RFIG

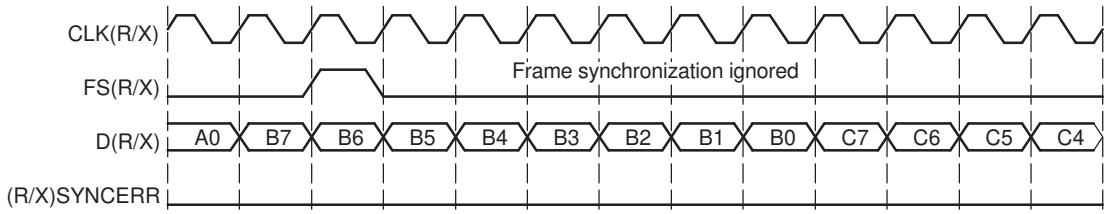
Figure 15-43 shows an example in which word B is interrupted by an unexpected frame-synchronization pulse when (R/X)FIG = 0. In the case of reception, the reception of B is aborted (B is lost), and a new data word (C in this example) is received after the appropriate data delay. This condition is a receive synchronization error, which sets the RSYNCERR bit.

Figure 15-43. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 0



In contrast with Figure 15-43, Figure 15-44 shows McBSP operation when unexpected frame-synchronization signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected pulse.

Figure 15-44. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 1



15.8.11 Set the Receive Companding Mode

The RCOMPAND bits (see Table 15-30) determine whether companding or another data transfer option is chosen for McBSP reception.

Table 15-30. Register Bits Used to Set the Receive Companding Mode

Register	Bit	Name	Function	Type	Reset Value
RCR2	4-3	RCOMPAND	Receive companding mode	R/W	00
			Modes other than 00b are enabled only when the appropriate RWDLEN is 000b, indicating 8-bit data.		
			RCOMPAND = 00 No companding, any size data, MSB received first		
			RCOMPAND = 01 No companding, 8-bit data, LSB received first (for details, see Section 15.8.11.4).		
			RCOMPAND = 10 μ-law companding, 8-bit data, MSB received first		
			RCOMPAND = 11 A-law companding, 8-bit data, MSB received first		

Copyright © 2012–2019, Texas Instruments Incorporated

15.8.11.1 Companding

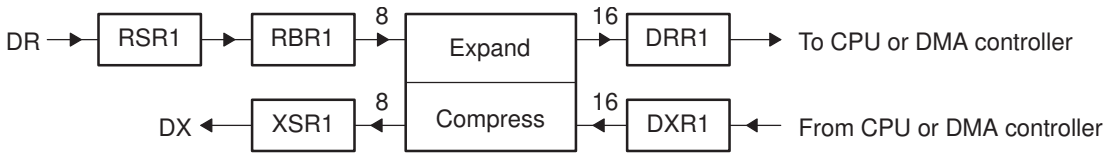
Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 15-45 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to 2's-complement format.

Figure 15-45. Companding Processes for Reception and for Transmission



15.8.11.2 Format of Expanded Data

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The RJUST bit of SPCR1 is ignored when companding is used.

15.8.11.3 Companding Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See Section 15.1.5.2, *Capability to Compand Internal Data*.

15.8.11.4 Option to Receive LSB First

Normally, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed during reception. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits and LSB-first ordering is done.

15.8.12 Set the Receive Data Delay

The RDATDLY bits (see Table 15-31) determine the length of the data delay for the receive frame.

Table 15-31. Register Bits Used to Set the Receive Data Delay

Register	Bit	Name	Function		Type	Reset Value
RCR2	1-0	RDATDLY	Receive data delay		R/W	00
			RDATDLY = 00	0-bit data delay		
			RDATDLY = 01	1-bit data delay		
			RDATDLY = 10	2-bit data delay		

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-31. Register Bits Used to Set the Receive Data Delay (continued)

Register	Bit	Name	Function	Type	Reset Value
			RDATDLY = 11 Reserved		

15.8.12.1 Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay.

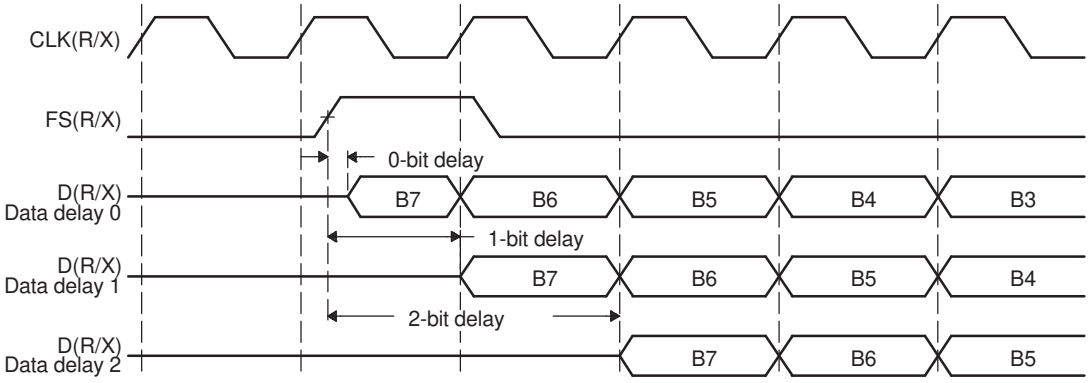
RDATDLY specifies the data delay for reception. The range of programmable data delay is zero to two bit-clocks (RDATDLY = 00b-10b), as described in Table 15-31 and shown in Figure 15-46. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1-cycle active frame-synchronization pulse.

15.8.12.2 0-Bit Data Delay

Normally, a frame-synchronization pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data may be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception, this problem is solved because receive data is sampled on the first falling edge of MCLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus on DX. The transmitter then asynchronously detects the frame-synchronization signal (FSX) going active high and immediately starts driving the first bit to be transmitted on the DX pin.

Figure 15-46. Range of Programmable Data Delay

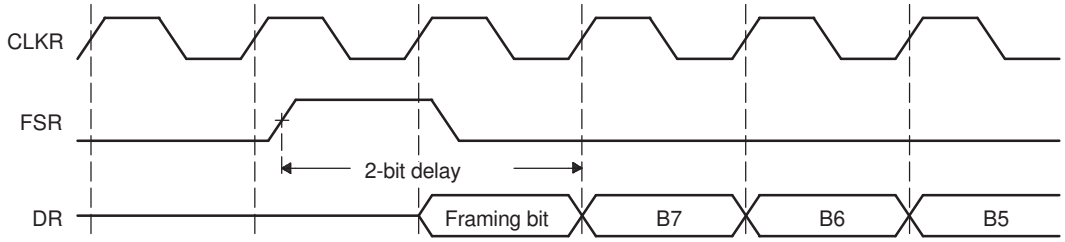


15.8.12.3 2-Bit Data Delay

A data delay of two bit periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in Figure 15-47. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-47. 2-Bit Data Delay Used to Skip a Framing Bit



15.8.13 Set the Receive Sign-Extension and Justification Mode

The RJUST bits (see Table 15-32) determine whether data received by the McBSP is sign-extended and how it is justified.

Table 15-32. Register Bits Used to Set the Receive Sign-Extension and Justification Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	14-13	RJUST	Receive sign-extension and justification mode		R/W	00
			RJUST = 00	Right justify data and zero fill MSBs in DRR[1,2]		
			RJUST = 01	Right justify data and sign extend it into the MSBs in DRR[1,2]		
			RJUST = 10	Left justify data and zero fill LSBs in DRR[1,2]		
			RJUST = 11	Reserved		

15.8.13.1 Sign-Extension and the Justification

RJUST in SPCR1 selects whether data in RBR[1,2] is right- or left-justified (with respect to the MSB) in DRR[1,2] and whether unused bits in DRR[1,2] are filled with zeros or with sign bits.

Table 15-33 and Table 15-34 show the effects of various RJUST values. The first table shows the effect on an example 12-bit receive-data value ABCh. The second table shows the effect on an example 20-bit receive-data value ABCDEh.

Table 15-33. Example: Use of RJUST Field With 12-Bit Data Value ABCh

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00b	Right	Zero fill MSBs	0000h	0ABCh
01b	Right	Sign extend data into MSBs	FFFFh	FABCh
10b	Left	Zero fill LSBs	0000h	ABC0h
11b	Reserved	Reserved	Reserved	Reserved

Table 15-34. Example: Use of RJUST Field With 20-Bit Data Value ABCDEh

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00b	Right	Zero fill MSBs	000Ah	BCDEh
01b	Right	Sign extend data into MSBs	FFFAh	BCDEh
10b	Left	Zero fill LSBs	ABCDh	E000h
11b	Reserved	Reserved	Reserved	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

15.8.14 Set the Receive Interrupt Mode

The RINTM bits (see Table 15-35) determine which event generates a receive interrupt request to the CPU.

The receive interrupt (RINT) informs the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the receive interrupt mode bits, RINTM, in SPCR1.

Table 15-35. Register Bits Used to Set the Receive Interrupt Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	5-4	RINTM	Receive interrupt mode		R/W	00
			RINTM = 00	RINT generated when RRDY changes from 0 to 1. Interrupt on every serial word by tracking the RRDY bit in SPCR1. Regardless of the value of RINTM, RRDY can be read to detect the RRDY = 1 condition.		
			RINTM = 01	RINT generated by an end-of-block or end-of-frame condition in the receive multichannel selection mode. In the multichannel selection mode, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. For details, see Section 15.6.7.3, *Using Interrupts Between Block Transfers*. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.		
			RINTM = 10	RINT generated by a new receive frame-synchronization pulse. Interrupt on detection of receive frame-synchronization pulses. This generates an interrupt even when the receiver is in its reset state. This is done by synchronizing the incoming frame-synchronization pulse to the CPU clock and sending it to the CPU via RINT.		
			RINTM = 11	RINT generated when RSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of RINTM, RSYNCERR can be read to detect this condition. For information on using RSYNCERR, see Section 15.5.3, *Unexpected Receive Frame-Synchronization Pulse*.		

15.8.15 Set the Receive Frame-Synchronization Mode

The bits described in Table 15-36 determine the source for receive frame synchronization and the function of the FSR pin.

15.8.15.1 Receive Frame-Synchronization Modes

Table 15-37 shows how you can select various sources to provide the receive frame-synchronization signal and the effect on the FSR pin. The polarity of the signal on the FSR pin is determined by the FSRP bit.

In digital loopback mode (DLB = 1), the transmit frame-synchronization signal is used as the receive frame-synchronization signal.

Also in the clock stop mode, the internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 15-36. Register Bits Used to Set the Receive Frame Synchronization Mode

Register	Bit	Name	Function		Type	Reset Value
PCR	10	FSRM	Receive frame-synchronization mode		R/W	0
			FSRM = 0	Receive frame synchronization is supplied by an external source via the FSR pin.		
			FSRM = 1	Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2.		

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-36. Register Bits Used to Set the Receive Frame Synchronization Mode (continued)

Register	Bit	Name	Function		Type	Reset Value
SRGR2	15	GSYNC	Sample rate generator clock synchronization mode		R/W	0
			If the sample rate generator creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin.			
			GSYNC = 0	No clock synchronization is used: CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.		
			GSYNC = 1	Clock synchronization is used. When a pulse is detected on the FSR pin: • CLKG is adjusted as necessary so that it is synchronized with the input clock on the MCLKR pin. • FSG pulses FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored. For more details, see Section 15.4.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.		
SPCR1	15	DLB	Digital loopback mode		R/W	0
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled. The receive signals, including the receive frame-synchronization signal, are connected internally through multiplexers to the corresponding transmit signals.		
SPCR1	12-11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; normal clocking for non-SPI mode.		
			CLKSTP = 10b	Clock stop mode enabled without clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		
			CLKSTP = 11b	Clock stop mode enabled with clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		

Table 15-37. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin

DLB	FSRM	GSYNC	Source of Receive Frame Synchronization	FSR Pin Status
0	0	0 or 1	An external frame-synchronization signal enters the McBSP through the FSR pin. The signal is then inverted as determined by FSRP before being used as internal FSR.	Input
0	1	0	Internal FSR is driven by the sample rate generator frame-synchronization signal (FSG).	Output. FSG is inverted as determined by FSRP before being driven out on the FSR pin.
0	1	1	Internal FSR is driven by the sample rate generator frame-synchronization signal (FSG).	Input. The external frame-synchronization input on the FSR pin is used to synchronize CLKG and generate FSG pulses.
1	0	0	Internal FSX drives internal FSR.	High impedance

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-37. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin (continued)

DLB	FSRM	GSYNC	Source of Receive Frame Synchronization	FSR Pin Status
1	0 or 1	1	Internal FSX drives internal FSR.	Input. If the sample rate generator is running, external FSR is used to synchronize CLKG and generate FSG pulses.
1	1	0	Internal FSX drives internal FSR.	Output. Receive (same as transmit) frame synchronization is inverted as determined by FSRP before being driven out on the FSR pin.

15.8.16 Set the Receive Frame-Synchronization Polarity

The FSRP bit (see Table 15-38) determines whether frame-synchronization pulses are active high or active low on the FSR pin.

Table 15-38. Register Bit Used to Set Receive Frame-Synchronization Polarity

Register	Bit	Name	Function		Type	Reset Value
PCR	2	FSRP	Receive frame-synchronization polarity		R/W	0
			FSRP = 0	Frame-synchronization pulse FSR is active high.		
			FSRP = 1	Frame-synchronization pulse FSR is active low.		

15.8.16.1 Frame-Synchronization Pulses, Clock Signals, and Their Polarities

Receive frame-synchronization pulses can be generated internally by the sample rate generator (see Section 15.4.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see Section 15.8.15, *Set the Receive Frame-Synchronization Mode*. Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR (see Section 15.8.17, *Set the Receive Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from an external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of the internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

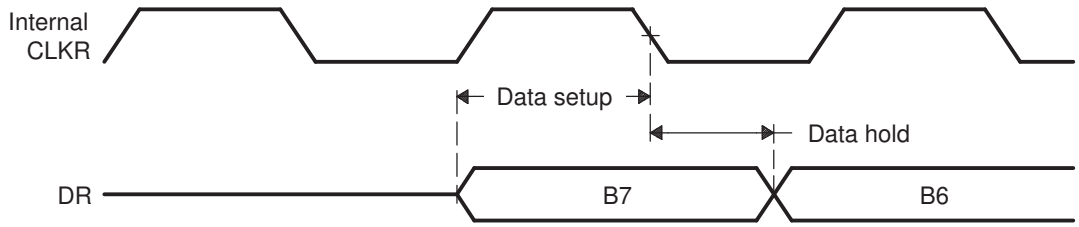
On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Copyright © 2012–2019, Texas Instruments Incorporated

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and MCLKR is an input pin), the external rising-edge triggered input clock on MCLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

MCLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 15-48 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 15-48. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



Set the SRG Frame-Synchronization Period and Pulse Width.

15.8.16.2 Frame-Synchronization Period and the Frame-Synchronization Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0. Table 15-39 shows settings for FPER and FWID.

Figure 15-49 shows a frame-synchronization period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

Table 15-39. Register Bits Used to Set the SRG Frame-Synchronization Period and Pulse Width

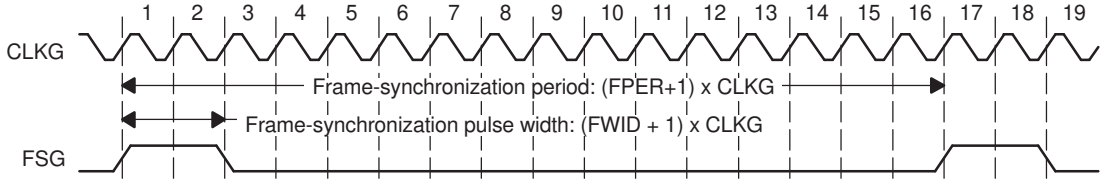
Register	Bit	Name	Function	Type	Reset Value
SRGR2	11-0	FPER	Sample rate generator frame-synchronization period For the frame-synchronization signal FSG, (FPER + 1) determines the period from the start of a frame-synchronization pulse to the start of the next frame-synchronization pulse. Range for (FPER + 1): 1 to 4096 CLKG cycles	R/W	0000 0000 0000
SRGR1	15-8	FWID	Sample rate generator frame-synchronization pulse width This field plus 1 determines the width of each frame-synchronization pulse on FSG.	R/W	0000 0000

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-39. Register Bits Used to Set the SRG Frame-Synchronization Period and Pulse Width (continued)

Register	Bit	Name	Function	Type	Reset Value
			Range for (FWID + 1): 1 to 256 CLKG cycles		

Figure 15-49. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when GRST = 1 and FSGM = 1, a frame-synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

15.8.17 Set the Receive Clock Mode

Table 15-40 shows the settings for bits used to set receive clock mode.

Table 15-40. Register Bits Used to Set the Receive Clock Mode

Register	Bit	Name	Function		Type	Reset Value
PCR	8	CLKRM	Receive clock mode		R/W	0
			Case 1: Digital loopback mode not set (DLB = 0) in SPCR1.			
			CLKRM = 0	The MCLKR pin is an input pin that supplies the internal receive clock (MCLKR).		
			CLKRM = 1	Internal MCLKR is driven by the sample rate generator of the McBSP. The MCLKR pin is an output pin that reflects internal MCLKR.		
			Case 2: Digital loopback mode set (DLB = 1) in SPCR1.			
			CLKRM = 0	The MCLKR pin is in the high impedance state. The internal receive clock (MCLKR) is driven by the internal transmit clock (CLKX). Internal CLKX is derived according to the CLKXM bit of PCR.		
			CLKRM = 1	Internal MCLKR is driven by internal CLKX. The MCLKR pin is an output pin that reflects internal MCLKR. Internal CLKX is derived according to the CLKXM bit of PCR.		
SPCR1	15	DLB	Digital loopback mode		R/W	00
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled. The receive signals, including the receive frame-synchronization signal, are connected internally through multiplexers to the corresponding transmit signals.		

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-40. Register Bits Used to Set the Receive Clock Mode (continued)

Register	Bit	Name	Function		Type	Reset Value
SPCR1	12-11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; normal clocking for non-SPI mode.		
			CLKSTP = 10b	Clock stop mode enabled without clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		
			CLKSTP = 11b	Clock stop mode enabled with clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.		

15.8.17.1 Selecting a Source for the Receive Clock and a Data Direction for the MCLKR Pin

Table 15-41 shows how you can select various sources to provide the receive clock signal and affect the MCLKR pin. The polarity of the signal on the MCLKR pin is determined by the CLKRP bit.

In the digital loopback mode (DLB = 1), the transmit clock signal is used as the receive clock signal.

Also, in the clock stop mode, the internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 15-41. Receive Clock Signal Source Selection

DLB in SPCR1	CLKRM in PCR	Source of Receive Clock	MCLKR Pin Status
0	0	The MCLKR pin is an input driven by an external clock. The external clock signal is inverted as determined by CLKRP before being used.	Input
0	1	The sample rate generator clock (CLKG) drives internal MCLKR.	Output. CLKG, inverted as determined by CLKRP, is driven out on the MCLKR pin.
1	0	Internal CLKX drives internal MCLKR. To configure CLKX, see Section 15.9.18, *Set the Transmit Clock Mode*.	High impedance
1	1	Internal CLKX drives internal MCLKR. To configure CLKX, see Section 15.9.18, *Set the Transmit Clock Mode*.	Output. Internal MCLKR (same as internal CLKX) is inverted as determined by CLKRP before being driven out on the MCLKR pin.

15.8.18 Set the Receive Clock Polarity

Table 15-42. Register Bit Used to Set Receive Clock Polarity

Register	Bit	Name	Function		Type	Reset Value
PCR	0	CLKRP	Receive clock polarity		R/W	0
			CLKRP = 0	Receive data sampled on falling edge of MCLKR		
			CLKRP = 1	Receive data sampled on rising edge of MCLKR		

Copyright © 2012–2019, Texas Instruments Incorporated

15.8.18.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Receive frame-synchronization pulses can be generated internally by the sample rate generator (see Section 15.4.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see Section 15.8.15, *Set the Receive Frame-Synchronization Mode*. Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR (see Section 15.8.17, *Set the Receive Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

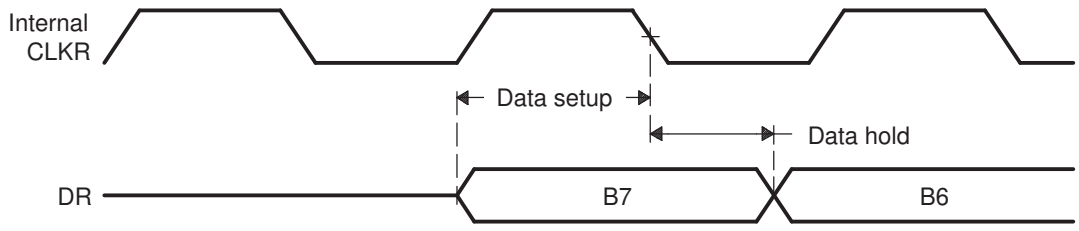
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and MCLKR is an input pin), the external rising-edge triggered input clock on MCLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 15-50 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 15-50. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



Copyright © 2012–2019, Texas Instruments Incorporated

15.8.19 Set the SRG Clock Divide-Down Value

Table 15-43. Register Bits Used to Set the Sample Rate Generator (SRG) Clock Divide-Down Value

Register	Bit	Name	Function	Type	Reset Value
SRGR1	7-0	CLKGDV	Sample rate generator clock divide-down value The input clock of the sample rate generator is divided by (CLKGDV + 1) to generate the required sample rate generator clock frequency. The default value of CLKGDV is 1 (divide input clock by 2).	R/W	0000 0001

15.8.19.1 Sample Rate Generator Clock Divider

The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV + 1) of sample rate generator input clock. Thus, the sample generator input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide-down, the high-state duration is p + 1 cycles and the low-state duration is p cycles.

15.8.20 Set the SRG Clock Synchronization Mode

For more details on using the clock synchronization feature, see Section 15.4.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.

Table 15-44. Register Bit Used to Set the SRG Clock Synchronization Mode

Register	Bit	Name	Function		Type	Reset Value
SRGR2	15	GSYNC	Sample rate generator clock synchronization GSYNC is used only when the input clock source for the sample rate generator is external—on the MCLKR or MCLKX pin.		R/W	0
			GSYNC = 0	The sample rate generator clock (CLKG) is free running. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.		
			GSYNC = 1	Clock synchronization is performed. When a pulse is detected on the FSR pin: • CLKG is adjusted as necessary so that it is synchronized with the input clock on the MCLKR or MCLKX pin. • FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored.		

15.8.21 Set the SRG Clock Mode (Choose an Input Clock)

Table 15-45. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock)

Register	Bit	Name	Function		Type	Reset Value
PCR	7	SCLKME	Sample rate generator clock mode		R/W	0
SRGR2	13	CLKSM			R/W	1
			SCLKME = 0 CLKSM = 0	Reserved		

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-45. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock) (continued)

Register	Bit	Name	Function		Type	Reset Value
			SCLKME = 0	Sample rate generator clock derived from LSPCLK (default)		
			CLKSM = 1			
			SCLKME = 1	Sample rate generator clock derived from MCLKR pin		
			CLKSM = 0			
			SCLKME = 1	Sample rate generator clock derived from MCLKX pin		
			CLKSM = 1			

15.8.21.1 SRG Clock Mode

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock. Table 15-45 shows the four possible sources of the input clock. For more details on generating CLKG, see Section 15.4.1.1, *Clock Generation in the Sample Rate Generator*.

Copyright © 2012–2019, Texas Instruments Incorporated

15.8.22 Set the SRG Input Clock Polarity

Table 15-46. Register Bits Used to Set the SRG Input Clock Polarity

Register	Bit	Name	Function		Type	Reset Value
PCR	1	CLKXP	MCLKX pin polarity		R/W	0
			CLKXP determines the input clock polarity when the MCLKX pin supplies the input clock (SCLKME = 1 and CLKSM = 1).			
			CLKXP = 0	Rising edge on MCLKX pin generates transitions on CLKG and FSG.		
			CLKXP = 1	Falling edge on MCLKX pin generates transitions on CLKG and FSG.		
PCR	0	CLKRP	MCLKR pin polarity		R/W	0
			CLKRP determines the input clock polarity when the MCLKR pin supplies the input clock (SCLKME = 1 and CLKSM = 0).			
			CLKRP = 0	Falling edge on MCLKR pin generates transitions on CLKG and FSG.		
			CLKRP = 1	Rising edge on MCLKR pin generates transitions on CLKG and FSG.		

15.8.22.1 Using CLKXP/CLKRP to Choose an Input Clock Polarity

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the CPU clock or from an external clock on the CLKX or MCLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKXP for the MCLKX pin, CLKRP for the MCLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

15.9 Transmitter Configuration

To configure the McBSP transmitter, perform the following procedure:

1. Place the McBSP/transmitter in reset (see Section 15.9.2).
2. Program the McBSP registers for the desired transmitter operation (see Section 15.9.1).
3. Take the transmitter out of reset (see Section 15.9.2).

15.9.1 Programming the McBSP Registers for the Desired Transmitter Operation

The following is a list of important tasks to be performed when you are configuring the McBSP transmitter. Each task corresponds to one or more McBSP register bit fields.

- Global behavior:
  - Set the transmitter pins to operate as McBSP pins.
  - Enable/disable the digital loopback mode.
  - Enable/disable the clock stop mode.
  - Enable/disable transmit multichannel selection.
- Data behavior:
  - Choose 1 or 2 phases for the transmit frame.
  - Set the transmit word length(s).
  - Set the transmit frame length.
  - Enable/disable the transmit frame-synchronization ignore function.
  - Set the transmit companding mode.
  - Set the transmit data delay.

Copyright © 2012–2019, Texas Instruments Incorporated

  - Set the transmit DXENA mode.
  - Set the transmit interrupt mode.
- Frame-synchronization behavior:
  - Set the transmit frame-synchronization mode.
  - Set the transmit frame-synchronization polarity.
  - Set the SRG frame-synchronization period and pulse width.
- Clock behavior:
  - Set the transmit clock mode.
  - Set the transmit clock polarity.
  - Set the SRG clock divide-down value.
  - Set the SRG clock synchronization mode.
  - Set the SRG clock mode (choose an input clock).
  - Set the SRG input clock polarity.

15.9.2 Resetting and Enabling the Transmitter

The first step of the transmitter configuration procedure is to reset the transmitter, and the last step is to enable the transmitter (to take it out of reset). Table 15-47 describes the bits used for both of these steps.

Table 15-47. Register Bits Used to Place Transmitter in Reset Field Descriptions

Register	Bit	Field	Value	Description
SPCR2	7	FRST		Frame-synchronization logic reset
			0	Frame-synchronization logic is reset. The sample rate generator does not generate frame-synchronization signal FSG, even if GRST = 1.
			1	Frame-synchronization is enabled. If GRST = 1, frame-synchronization signal FSG is generated after (FPER + 1) number of CLKG clock cycles; all frame counters are loaded with their programmed values.
SPCR2	6	GRST		Sample rate generator reset
			0	Sample rate generator is reset. If GRST = 0 due to a device reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 due to program code, CLKG and FSG are both driven low (inactive).
			1	Sample rate generator is enabled. CLKG is driven according to the configuration programmed in the sample rate generator registers (SRGR[1,2]). If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers.
SPCR2	0	XRST		Transmitter reset
			0	The serial port transmitter is disabled and in the reset state.
			1	The serial port transmitter is enabled.

15.9.2.1 Reset Considerations

The serial port can be reset in the following two ways:

1. A DSP reset ($\overline{\text{XRS}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed, GRST = FRST = RRST = XRST = 0, keeping the entire serial port in the reset state.
2. The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the serial port control registers. The sample rate generator can be reset directly using the GRST bit in SPCR2.
3. When using the DMA, the order in which McBSP events must occur is important. DMA channel and peripheral interrupts must be configured prior to releasing the McBSP transmitter from reset.

   The reason for this is that an XRDY is fired when XRST = 1. The XRDY signals the DMA to start copying data from the buffer into the transmit register. If the McBSP transmitter is released from reset before the DMA channel and peripheral interrupts are configured, the XRDY signals before the DMA channel can receive the signal; therefore, the DMA does not move the data from the buffer to the

Copyright © 2012–2019, Texas Instruments Incorporated

transmit register. The DMA PERINTFLG is edge-sensitive and will fail to recognize the XRDY, which is continuously high.

For more details about McBSP reset conditions and effects, see Section 15.10.2, *Resetting and Initializing a McBSP*.

15.9.3 Set the Transmitter Pins to Operate as McBSP Pins

To configure a pin for its McBSP function , you should configure the bits of the GPxMUXn register appropriately. In addition to this, bits 12 and 13 of the PCR register must be set to 0. These bits are defined as reserved.

15.9.4 Enable/Disable the Digital Loopback Mode

The DLB bit determines whether the digital loopback mode is on. DLB is described in Table 15-48.

Table 15-48. Register Bit Used to Enable/Disable the Digital Loopback Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	15	DLB	Digital loopback mode		R/W	0
			DLB = 0	Digital loopback mode is disabled.		
			DLB = 1	Digital loopback mode is enabled.		

15.9.4.1 Digital Loopback Mode

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in Table 15-49. This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

Table 15-49. Receive Signals Connected to Transmit Signals in Digital Loopback Mode

This Receive Signal	Is Fed Internally by This Transmit Signal
DR (receive data)	DX (transmit data)
FSR (receive frame synchronization)	FSX (transmit frame synchronization)
MCLKR (receive clock)	CLKX (transmit clock)

15.9.5 Enable/Disable the Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on. CLKSTP is described in Table 15-50.

Table 15-50. Register Bits Used to Enable/Disable the Clock Stop Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	12-11	CLKSTP	Clock stop mode		R/W	00
			CLKSTP = 0Xb	Clock stop mode disabled; normal clocking for non-SPI mode.		
			CLKSTP = 10b	Clock stop mode enabled without clock delay		
			CLKSTP = 11b	Clock stop mode enabled with clock delay		

Copyright © 2012–2019, Texas Instruments Incorporated

15.9.5.1 Clock Stop Mode

The clock stop mode supports the SPI master-slave protocol. If you do not plan to use the SPI protocol, you can clear CLKSTP to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). The CLKXP bit determines whether the starting edge of the clock on the MCLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the MCLKR pin.

Table 15-51 summarizes the impact of CLKSTP, CLKXP, and CLKRP on serial port operation. In the clock stop mode, the receive clock is tied internally to the transmit clock, and the receive frame-synchronization signal is tied internally to the transmit frame-synchronization signal.

Table 15-51. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of MCLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of MCLKR.

Copyright © 2012–2019, Texas Instruments Incorporated

15.9.6 Enable/Disable Transmit Multichannel Selection

For more details, see Section 15.6.7, *Transmit Multichannel Selection Modes*.

Table 15-52. Register Bits Used to Enable/Disable Transmit Multichannel Selection

Register	Bit	Name	Function		Type	Reset Value
MCR2	1-0	XMCM	Transmit multichannel selection		R/W	00
			XMCM = 00b	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.		
			XMCM = 01b	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked.		
				The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.		
			XMCM = 10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs).		
				The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.		
			XMCM = 11b	This mode is used for symmetric transmission and reception.		
				All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs).		
				The XMCME bit determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.		

15.9.7 Choose One or Two Phases for the Transmit Frame

Table 15-53. Register Bit Used to Choose 1 or 2 Phases for the Transmit Frame

Register	Bit	Name	Function		Type	Reset Value
XCR2	15	XPHASE	Transmit phase number		R/W	0
			Specifies whether the transmit frame has 1 or 2 phases.			
			XPHASE = 0	Single-phase frame		
			XPHASE = 1	Dual-phase frame		

15.9.8 Set the Transmit Word Length(s)

Table 15-54. Register Bits Used to Set the Transmit Word Length(s)

Register	Bit	Name	Function		Type	Reset Value
XCR1	7-5	XWDLEN1	Transmit word length of frame phase 1		R/W	000
			XWDLEN1 = 000b	8 bits		
			XWDLEN1 = 001b	12 bits		
			XWDLEN1 = 010b	16 bits		
			XWDLEN1 = 011b	20 bits		
			XWDLEN1 = 100b	24 bits		
			XWDLEN1 = 101b	32 bits		
			XWDLEN1 = 11Xb	Reserved		

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-54. Register Bits Used to Set the Transmit Word Length(s) (continued)

Register	Bit	Name	Function		Type	Reset Value
XCR2	7-5	XWDLEN2	Transmit word length of frame phase 2		R/W	000
			XWDLEN2 = 000b	8 bits		
			XWDLEN2 = 001b	12 bits		
			XWDLEN2 = 010b	16 bits		
			XWDLEN2 = 011b	20 bits		
			XWDLEN2 = 100b	24 bits		
			XWDLEN2 = 101b	32 bits		
			XWDLEN2 = 11Xb	Reserved		

15.9.8.1 Word Length Bits

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, XWDLEN1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame, and XWDLEN2 determines the word length in phase 2 of the frame.

Copyright © 2012–2019, Texas Instruments Incorporated

15.9.9 Set the Transmit Frame Length

Table 15-55. Register Bits Used to Set the Transmit Frame Length

Register	Bit	Name	Function		Type	Reset Value
XCR1	14-8	XFRLEN1	Transmit frame length 1		R/W	000 0000
			(XFRLEN1 + 1) is the number of serial words in phase 1 of the transmit frame.			
			XFRLEN1 = 000 0000	1 word in phase 1		
			XFRLEN1 = 000 0001	2 words in phase 1		
			|	|		
			|	|		
			XFRLEN1 = 111 1111	128 words in phase 1		
XCR2	14-8	XFRLEN2	Transmit frame length 2		R/W	000 0000
			If a dual-phase frame is selected, (XFRLEN2 + 1) is the number of serial words in phase 2 of the transmit frame.			
			XFRLEN2 = 000 0000	1 word in phase 2		
			XFRLEN2 = 000 0001	2 words in phase 2		
			|	|		
			|	|		
			XFRLEN2 = 111 1111	128 words in phase 2		

15.9.9.1 Selected Frame Length

The transmit frame length is the number of serial words in the transmit frame. Each frame can have one or two phases, depending on the value that you load into the XPHASE bit.

If a single-phase frame is selected (XPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (XPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 15-56 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

NOTE: Program the XFRLEN fields with [*w minus 1*], where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.

Table 15-56. How to Calculate Frame Length

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 ≤ XFRLEN1 ≤ 127	Don't care	(XFRLEN1 + 1) words
1	0 ≤ XFRLEN1 ≤ 127	0 ≤ XFRLEN2 ≤ 127	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

Copyright © 2012–2019, Texas Instruments Incorporated

15.9.10 Enable/Disable the Transmit Frame-Synchronization Ignore Function

Table 15-57. Register Bit Used to Enable/Disable the Transmit Frame-Synchronization Ignore Function

Register	Bit	Name	Function		Type	Reset Value
XCR2	2	XFIG	Transmit frame-synchronization ignore		R/W	0
			XFIG = 0	An unexpected transmit frame-synchronization pulse causes the McBSP to restart the frame transfer.		
			XFIG = 1	The McBSP ignores unexpected transmit frame-synchronization pulses.		

15.9.10.1 Unexpected Frame-Synchronization Pulses and Frame-Synchronization Ignore

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse.

When XFIG = 1, normal transmission continues with unexpected frame-synchronization signals ignored.

When XFIG = 0 and an unexpected frame-synchronization pulse occurs, the serial port:

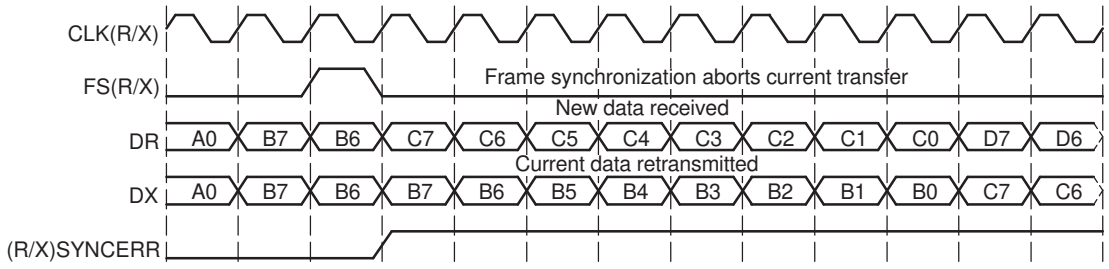
1. Aborts the present transmission
2. Sets XSYNCERR to 1 in SPCR2
3. Reinitiates transmission of the current word that was aborted

For more details about the frame-synchronization error condition, see Section 15.5.5, *Unexpected Transmit Frame-Synchronization Pulse*.

15.9.10.2 Examples Showing the Effects of XFIG

Figure 15-51 shows an example in which word B is interrupted by an unexpected frame-synchronization pulse when (R/X)FIG = 0. In the case of transmission, the transmission of B is aborted (B is lost). This condition is a transmit synchronization error, which sets the XSYNCERR bit. No new data has been written to DXR[1,2]; therefore, the McBSP transmits B again.

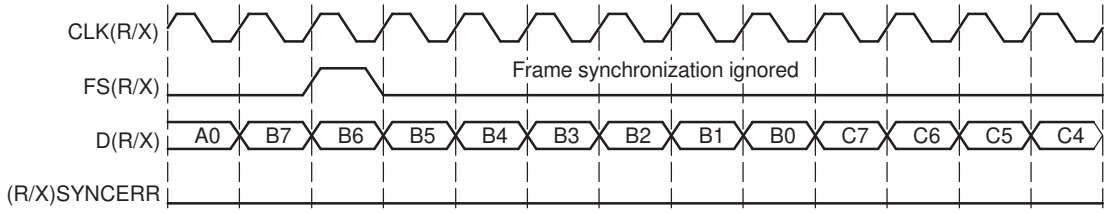
Figure 15-51. Unexpected Frame-Synchronization Pulse With (R/X) FIG = 0



In contrast with Figure 15-51, Figure 15-52 shows McBSP operation when unexpected frame-synchronization signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected frame-synchronization pulse.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-52. Unexpected Frame-Synchronization Pulse With (R/X) FIG = 1



15.9.11 Set the Transmit Companding Mode

Table 15-58. Register Bits Used to Set the Transmit Companding Mode

Register	Bit	Name	Function		Type	Reset Value
XCR2	4-3	XCOMPAND	Transmit companding mode		R/W	00
			Modes other than 00b are enabled only when the appropriate XWDLEN is 000b, indicating 8-bit data.			
			XCOMPAND = 00b	No companding, any size data, MSB transmitted first		
			XCOMPAND = 01b	No companding, 8-bit data, LSB transmitted first (for details, see Section 15.8.11.4, *Option to Receive LSB First*)		
			XCOMPAND = 10b	μ-law companding, 8-bit data, MSB transmitted first		
			XCOMPAND = 11b	A-law companding, 8-bit data, MSB transmitted first		

15.9.11.1 Companding

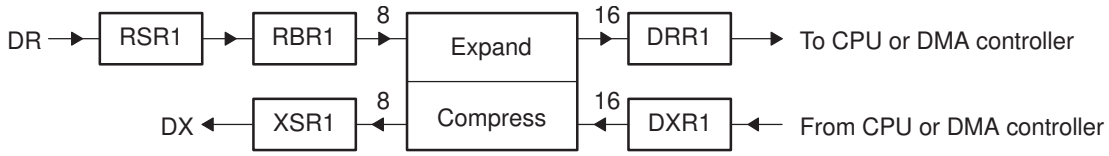
Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 15-53 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to twos-complement format.

Figure 15-53. Companding Processes for Reception and for Transmission

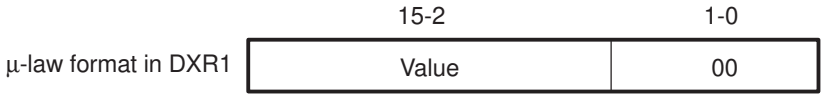


Copyright © 2012–2019, Texas Instruments Incorporated

15.9.11.2 Format for Data To Be Compressed

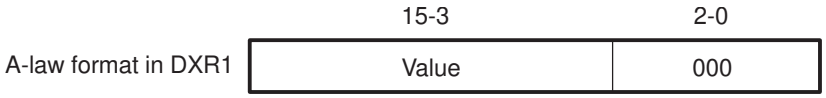
For transmission using μ-law compression, make sure the 14 data bits are left-justified in DXR1, with the remaining two low-order bits filled with 0s as shown in Figure 15-54.

Figure 15-54. μ-Law Transmit Data Companding Format



For transmission using A-law compression, make sure the 13 data bits are left-justified in DXR1, with the remaining three low-order bits filled with 0s as shown in Figure 15-55.

Figure 15-55. A-Law Transmit Data Companding Format



15.9.11.3 Capability to Compand Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See Section 15.1.5.2, *Capability to Compand Internal Data*.

15.9.11.4 Option to Transmit LSB First

Normally, the McBSP transmit or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits and LSB-first ordering is done.

15.9.12 Set the Transmit Data Delay

Table 15-59. Register Bits Used to Set the Transmit Data Delay

Register	Bit	Name	Function		Type	Reset Value
XCR2	1-0	XDATDLY	Transmitter data delay		R/W	00
			XDATDLY = 00	0-bit data delay		
			XDATDLY = 01	1-bit data delay		
			XDATDLY = 10	2-bit data delay		
			XDATDLY = 11	Reserved		

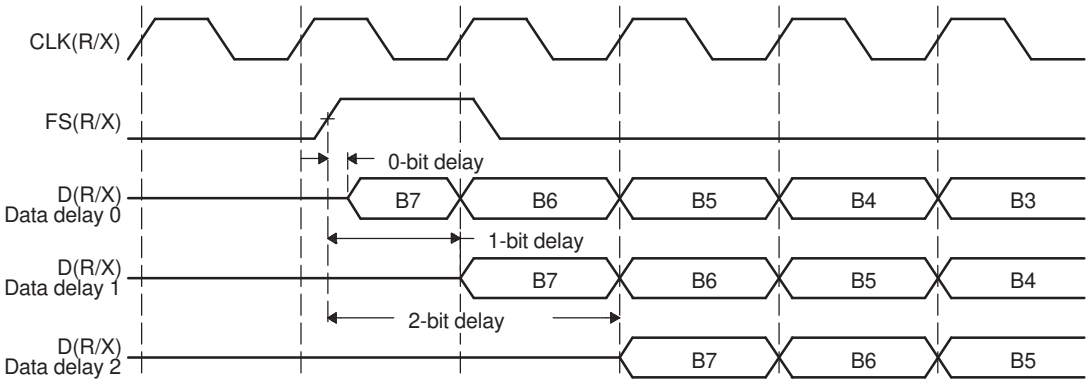
15.9.12.1 Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if necessary. This delay is called data delay.

XDATDLY specifies the data delay for transmission. The range of programmable data delay is zero to two bit-clocks (XDATDLY = 00b-10b), as described in Table 15-59 and Figure 15-56. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1-cycle active frame-synchronization pulse.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-56. Range of Programmable Data Delay



15.9.12.2 0-Bit Data Delay

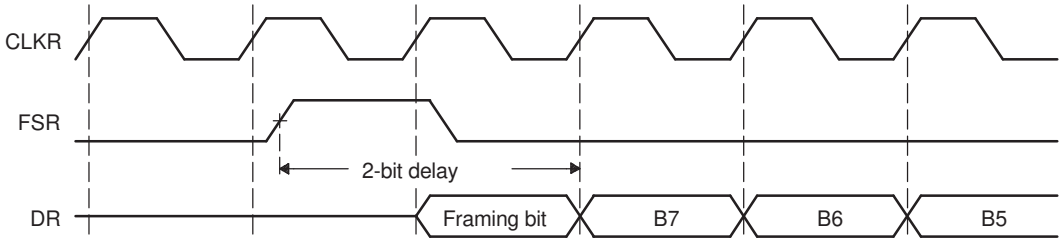
Normally, a frame-synchronization pulse is detected or sampled with respect to an edge of serial clock internal CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data can be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception this problem is solved because receive data is sampled on the first falling edge of MCLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus DX. The transmitter then asynchronously detects the frame synchronization, FSX, going active high and immediately starts driving the first bit to be transmitted on the DX pin.

15.9.12.3 2-Bit Data Delay

A data delay of two bit-periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in the following figure. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 15-57. 2-Bit Data Delay Used to Skip a Framing Bit



Copyright © 2012–2019, Texas Instruments Incorporated

15.9.13 Set the Transmit DXENA Mode

Table 15-60. Register Bit Used to Set the Transmit DXENA (DX Delay Enabler) Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR1	7	DXENA	DX delay enabler mode		R/W	0
			DXENA = 0	DX delay enabler is off.		
			DXENA = 1	DX delay enabler is on.		

15.9.13.1 DXENA Mode

The DXENA bit controls the delay enabler on the DX pin. Set DXENA to enable an extra delay for turn-on time. This bit does not control the data itself, so only the first bit is delayed.

If you tie together the DX pins of multiple McBSPs, make sure DXENA = 1 to avoid having more than one McBSP transmit on the data line at one time.

15.9.14 Set the Transmit Interrupt Mode

The transmitter interrupt (XINT) signals the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the transmit interrupt mode bits, XINTM, in SPCR2.

Table 15-61. Register Bits Used to Set the Transmit Interrupt Mode

Register	Bit	Name	Function		Type	Reset Value
SPCR2	5-4	XINTM	Transmit interrupt mode		R/W	00
			XINTM = 00	XINT generated when XRDY changes from 0 to 1.		
			XINTM = 01	XINT generated by an end-of-block or end-of-frame condition in a transmit multichannel selection mode. In any of the transmit multichannel selection modes, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. For details, see Section 15.6.7.3, *Using Interrupts Between Block Transfers*. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.		
			XINTM = 10	XINT generated by a new transmit frame-synchronization pulse. Interrupt on detection of each transmit frame-synchronization pulse. This generates an interrupt even when the transmitter is in its reset state. This is done by synchronizing the incoming frame-synchronization pulse to the CPU clock and sending it to the CPU via XINT.		
			XINTM = 11	XINT generated when XSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of XINTM, XSYNCERR can be read to detect this condition. For more information on using XSYNCERR, see Section 15.5.5, *Unexpected Transmit Frame-Synchronization Pulse*.		

Copyright © 2012–2019, Texas Instruments Incorporated

15.9.15 Set the Transmit Frame-Synchronization Mode

Table 15-62. Register Bits Used to Set the Transmit Frame-Synchronization Mode

Register	Bit	Name	Function		Type	Reset Value
PCR	11	FSXM	Transmit frame-synchronization mode		R/W	0
			FSXM = 0	Transmit frame synchronization is supplied by an external source via the FSX pin.		
			FSXM = 1	Transmit frame synchronization is supplied by the McBSP, as determined by the FSGM bit of SRGR2.		
SRGR2	12	FSGM	Sample rate generator transmit frame-synchronization mode		R/W	0
			Used when FSXM = 1 in PCR.			
			FSGM = 0	The McBSP generates a transmit frame-synchronization pulse when the content of DXR[1,2] is copied to XSR[1,2].		
			FSGM = 1	The transmitter uses frame-synchronization pulses generated by the sample rate generator. Program the FWID bits to set the width of each pulse. Program the FPER bits to set the frame-synchronization period.		

15.9.15.1 Transmit Frame-Synchronization Modes

Table 15-63 shows how FSXM and FSGM select the source of transmit frame-synchronization pulses. The three choices are:

- External frame-synchronization input
- Sample rate generator frame-synchronization signal (FSG)
- Internal signal that indicates a DXR-to-XSR copy has been made

Table 15-63 also shows the effect of each bit setting on the FSX pin. The polarity of the signal on the FSX pin is determined by the FSXP bit.

Table 15-63. How FSXM and FSGM Select the Source of Transmit Frame-Synchronization Pulses

FSXM	FSGM	Source of Transmit Frame Synchronization	FSX Pin Status
0	0 or 1	An external frame-synchronization signal enters the McBSP through the FSX pin. The signal is then inverted by FSXP before being used as internal FSX.	Input
1	1	Internal FSX is driven by the sample rate generator frame-synchronization signal (FSG).	Output. FSG is inverted by FSXP before being driven out on FSX pin.
1	0	A DXR-to-XSR copy causes the McBSP to generate a transmit frame-synchronization pulse that is 1 cycle wide.	Output. The generated frame-synchronization pulse is inverted as determined by FSXP before being driven out on FSX pin.

15.9.15.2 Other Considerations

If the sample rate generator creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin. For more details, see Section 15.4.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master and must provide a slave-enable signal ($\overline{\text{SPISTE}}$) on the FSX pin, make sure that FSXM = 1 and FSGM = 0 so that FSX is an output and is driven active for the duration of each transmission. If the McBSP is a slave, make sure that FSXM = 0 so that the McBSP can receive the slave-enable signal on the FSX pin.

Copyright © 2012–2019, Texas Instruments Incorporated

15.9.16 Set the Transmit Frame-Synchronization Polarity

Table 15-64. Register Bit Used to Set Transmit Frame-Synchronization Polarity

Register	Bit	Name	Function		Type	Reset Value
PCR	3	FSXP	Transmit frame-synchronization polarity		R/W	0
			FSXP = 0	Frame-synchronization pulse FSX is active high.		
			FSXP = 1	Frame-synchronization pulse FSX is active low.		

15.9.16.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Transmit frame-synchronization pulses can be generated internally by the sample rate generator (see Section 15.4.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see Section 15.9.15, *Set the Transmit Frame-Synchronization Mode*). Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see Section 15.9.18, *Set the Transmit Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected and the polarity bit FS(R/X)P = 1, the internal active-high frame-synchronization signals are inverted before being sent to the FS(R/X) pin.

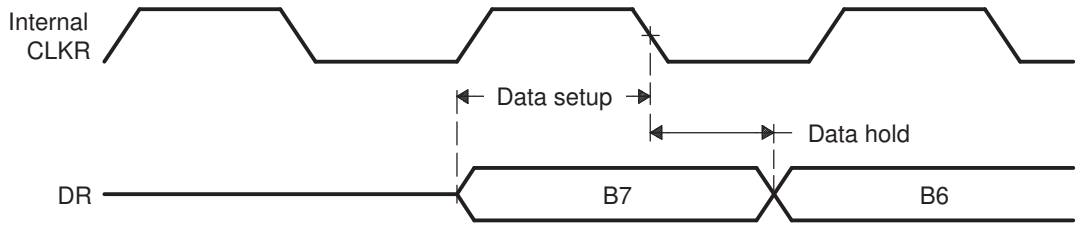
On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and MCLKR is an input pin), the external rising-edge triggered input clock on MCLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 15-58 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-58. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



15.9.17 Set the SRG Frame-Synchronization Period and Pulse Width

Table 15-65. Register Bits Used to Set SRG Frame-Synchronization Period and Pulse Width

Register	Bit	Name	Function	Type	Reset Value
SRGR2	11-0	FPER	Sample rate generator frame-synchronization period	R/W	0000 0000 0000
			For the frame-synchronization signal FSG, (FPER + 1) determines the period from the start of a frame-synchronization pulse to the start of the next frame-synchronization pulse.		
			Range for (FPER + 1): 1 to 4096 CLKG cycles.		
SRGR1	15-8	FWID	Sample rate generator frame-synchronization pulse width	R/W	0000 0000
			This field plus 1 determines the width of each frame-synchronization pulse on FSG.		
			Range for (FWID + 1): 1 to 256 CLKG cycles.		

15.9.17.1 Frame-Synchronization Period and Frame-Synchronization Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

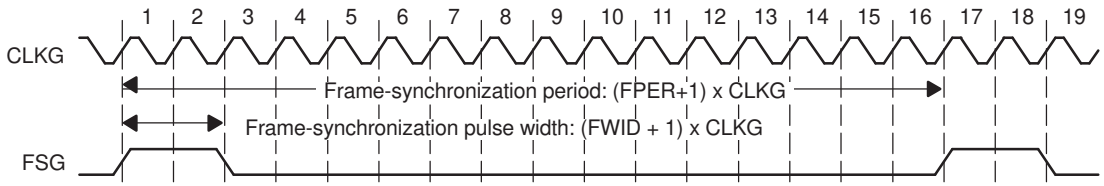
On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0.

Figure 15-59 shows a frame-synchronization period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

Figure 15-59. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



Copyright © 2012–2019, Texas Instruments Incorporated

When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when GRST = 1 and FSGM = 1, a frame-synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

15.9.18 Set the Transmit Clock Mode

Table 15-66. Register Bit Used to Set the Transmit Clock Mode

Register	Bit	Name	Function		Type	Reset Value
PCR	9	CLKXM	Transmit clock mode		R/W	0
			CLKXM = 0	The transmitter gets its clock signal from an external source via the MCLKX pin.		
			CLKXM = 1	The MCLKX pin is an output pin driven by the sample rate generator of the McBSP.		

15.9.18.1 Selecting a Source for the Transmit Clock and a Data Direction for the MCLKX pin

Table 15-67 shows how the CLKXM bit selects the transmit clock and the corresponding status of the MCLKX pin. The polarity of the signal on the MCLKX pin is determined by the CLKXP bit.

Table 15-67. How the CLKXM Bit Selects the Transmit Clock and the Corresponding Status of the MCLKX pin

CLKXM in PCR	Source of Transmit Clock	MCLKX pin Status
0	Internal CLKX is driven by an external clock on the MCLKX pin. CLKX is inverted as determined by CLKXP before being used.	Input
1	Internal CLKX is driven by the sample rate generator clock, CLKG.	Output. CLKG, inverted as determined by CLKXP, is driven out on CLKX.

15.9.18.2 Other Considerations

If the sample rate generator creates a clock signal (CLKG) that is derived from an external input clock, the GSYNC bit determines whether CLKG is kept synchronized with pulses on the FSR pin. For more details, see Section 15.4.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master, make sure that CLKXM = 1 so that CLKX is an output to supply the master clock to any slave devices. If the McBSP is a slave, make sure that CLKXM = 0 so that CLKX is an input to accept the master clock signal.

15.9.19 Set the Transmit Clock Polarity

Table 15-68. Register Bit Used to Set Transmit Clock Polarity

Register	Bit	Name	Function		Type	Reset Value
PCR	1	CLKXP	Transmit clock polarity		R/W	0
			CLKXP = 0	Transmit data sampled on rising edge of CLKX.		
			CLKXP = 1	Transmit data sampled on falling edge of CLKX.		

Copyright © 2012–2019, Texas Instruments Incorporated

15.9.19.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Transmit frame-synchronization pulses can be either generated internally by the sample rate generator (see Section 15.4.2) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see Section 15.9.15, *Set the Transmit Frame-Synchronization Mode*). Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see Section 15.9.18, *Set the Transmit Clock Mode*).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

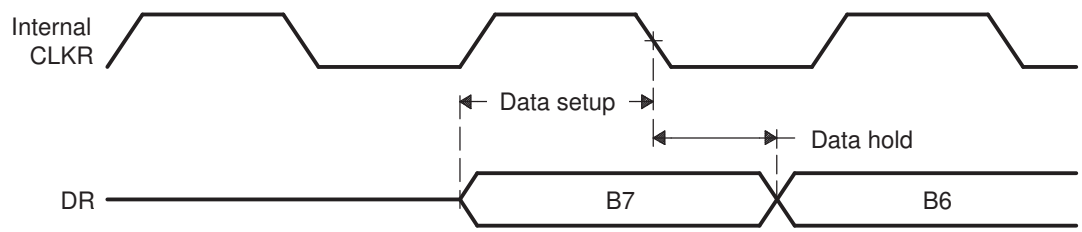
On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge (see Figure 15-58).

Figure 15-60 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 15-60. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



Copyright © 2012–2019, Texas Instruments Incorporated

15.10 Emulation and Reset Considerations

This section covers the following topics:

- How to program McBSP response to a breakpoint in the high-level language debugger (see Section 15.10.1)
- How to reset and initialize the various parts of the McBSP (see Section 15.10.2)

15.10.1 McBSP Emulation Mode

FREE and SOFT are special emulation bits in SPCR2 that determine the state of the McBSP when a breakpoint is encountered in the high-level language debugger. If FREE = 1, the clock continues to run upon a software breakpoint and data is still shifted out. When FREE = 1, the SOFT bit is a *don't care.*

If FREE = 0, the SOFT bit takes effect. If SOFT = 0 when breakpoint occurs, the clock stops immediately, aborting a transmission. If SOFT = 1 and a breakpoint occurs while transmission is in progress, the transmission continues until completion of the transfer and then the clock halts. These options are listed in Table 15-69.

The McBSP receiver functions in a similar fashion. If a mode other than the immediate stop mode (SOFT = FREE = 0) is chosen, the receiver continues running and an overrun error is possible.

Table 15-69. McBSP Emulation Modes Selectable with FREE and SOFT Bits of SPCR2

FREE	SOFT	McBSP Emulation Mode
0	0	Immediate stop mode (reset condition)
		The transmitter or receiver stops immediately in response to a breakpoint.
0	1	Soft stop mode
		When a breakpoint occurs, the transmitter stops after completion of the current word. The receiver is not affected.
1	0 or 1	Free run mode
		The transmitter and receiver continue to run when a breakpoint occurs.

15.10.2 Resetting and Initializing McBSP

15.10.2.1 McBSP Pin States: DSP Reset Versus Receiver/Transmitter Reset

Table 15-70 shows the state of McBSP pins when the serial port is reset due to direct receiver or transmitter reset on the device.

Table 15-70. Reset State of Each McBSP Pin

Pin	Possible State(s)[1]	State Forced by Device Reset	State Forced by Receiver/Transmitter Reset
Receiver reset (RRST = 0 and GRST = 1)			
MDRx	I	GPIO-input	Input
MCLKRx	I/O/Z	GPIO-input	Known state if input; MCLKR running if output
MFSRx	I/O/Z	GPIO-input	Known state if input; FSRP inactive state if output
Transmitter reset (XRST = 0 and GRST = 1)			
MDXx	O/Z	GPIO Input	High impedance
MCLKXx	I/O/Z	GPIO-input	Known state if input; CLKX running if output
MFSXx	I/O/Z	GPIO-input	Known state if input; FSXP inactive state if output

(1) In Possible State(s) column, I = Input, O = Output, Z = High impedance. In the 28x family, at device reset, all I/Os default to GPIO function and generally as inputs.

Copyright © 2012–2019, Texas Instruments Incorporated

15.10.2.2 Device Reset, McBSP Reset, and Sample Rate Generator Reset

When the McBSP is reset in either of the above two ways, the machine is reset to its initial state, including reset of all counters and status bits. The receive status bits include RFULL, RRDY, and RSYNCERR. The transmit status bits include XEMPTY, XRDY, and XSYNCERR.

- Device reset. When the whole DSP is reset ($\overline{\text{XRS}}$ signal is driven low), all McBSP pins are in GPIO mode. When the device is pulled out of reset, the clock to the McBSP modules remains disabled.
- McBSP reset. When the receiver and transmitter reset bits, RRST and XRST, are loaded with 0s, the respective portions of the McBSP are reset and activity in the corresponding section of the serial port stops. Input-only pins such as MDRx, and all other pins that are configured as inputs are in a known state. The MFSRx and MFSXx pins are driven to their inactive state if they are not outputs. If the MCLKR and MCLKX pins are programmed as outputs, they are driven by CLKG, provided that GRST = 1. Lastly, the MDXx pin is in the high-impedance state when the transmitter and/or the device is reset.

  During normal operation, the sample rate generator is reset if the GRST bit is cleared. GRST must be 0 only when neither the transmitter nor the receiver is using the sample rate generator. In this case, the internal sample rate generator clock (CLKG) and its frame-synchronization signal (FSG) are driven inactive low.

  When the sample rate generator is not in the reset state (GRST = 1), pins MFSRx and MFSXx are in an inactive state when RRST = 0 and XRST = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the McBSP is in reset, the other portion can continue operation when GRST = 1 and its frame synchronization is driven by FSG.
- Sample rate generator reset. The sample rate generator is reset when GRST is loaded with 0.

  When neither the transmitter nor the receiver is fed by CLKG and FSG, you can reset the sample rate generator by clearing GRST. In this case, CLKG and FSG are driven inactive low. If you then set GRST, CLKG starts and runs as programmed. Later, if GRST = 1, FSG pulses active high after the programmed number of CLKG cycles has elapsed.

15.10.2.3 McBSP Initialization Procedure

The serial port initialization procedure is as follows:

1. Make XRST = RRST = GRST = 0 in SPCR[1,2]. If coming out of a device reset, this step is not required.
2. While the serial port is in the reset state, program only the McBSP configuration registers (not the data registers) as required.
3. Wait for two clock cycles. This ensures proper internal synchronization.
4. Set up data acquisition as required (such as writing to DXR[1,2]).
5. Make XRST = RRST = 1 to enable the serial port. Make sure that as you set these reset bits, you do not modify any of the other bits in SPCR1 and SPCR2. Otherwise, you change the configuration you selected in step 2.
6. Set FRST = 1, if internally generated frame synchronization is required.
7. Wait two clock cycles for the receiver and transmitter to become active.

Alternatively, on either write (step 1 or 5), the transmitter and receiver can be placed in or taken out of reset individually by modifying the desired bit.

The above procedure for reset/initialization can be applied in general when the receiver or transmitter must be reset during its normal operation and when the sample rate generator is not used for either operation.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE:

1. The necessary duration of the active-low period of XRST or RRST is at least two MCLKR/CLKX cycles.
2. The appropriate bits in serial port configuration registers SPCR[1,2], PCR, RCR[1,2], XCR[1,2], and SRGR[1,2] must only be modified when the affected portion of the serial port is in its reset state.
3. In most cases, the data transmit registers (DXR[1,2]) must be loaded by the CPU or by the DMA controller only when the transmitter is enabled (XRST = 1). An exception to this rule is when these registers are used for companding internal data (see Section 15.1.5.2
4. Section 15.1.5.2, *Capability to Compand Internal Data*).
5. The bits of the channel control registers—MCR[1,2], RCER[A-H], XCER[A-H]—can be modified at any time as long as they are not being used by the current reception/transmission in a multichannel selection mode.

15.10.2.4 Resetting the Transmitter While the Receiver is Running

Example 5 shows values in the control registers that reset and configure the transmitter while the receiver is running.

Equation 5: Resetting and Configuring McBSP Transmitter While the McBSP Receiver Running

```
SPCR1 = 0001h SPCR2 = 0030h
; The receiver is running with the receive interrupt (RINT) triggered by the
; receiver ready bit (RRDY). The transmitter is in its reset state
. The transmit interrupt (XINT) will be triggered by the transmit frame-sync
; error bit (XSYNCERR). PCR = 0900h
; Transmit frame synchronization is generated internally according to the
; FSGM bit of SRGR2.
; The transmit clock is driven by an external source.
; The receive clock continues to be driven by sample rate generator. The input clock
; of the sample rate generator is supplied by the CPU clock SRGR1 = 0001h SRGR2 = 2000h
; The CPU clock is the input clock for the sample rate generator. The sample
; rate generator divides the CPU clock by 2 to generate its output clock (CLKG).
; Transmit frame synchronization is tied to the automatic copying of data from
; the DXR(s) to the XSR(s). XCR1 = 0740h XCR2 = 8321h
; The transmit frame has two phases. Phase 1 has eight 16-bit words. Phase 2
; has four 12-bit words. There is 1-bit data delay between the start of a
; frame-sync pulse and the first data bit
; transmitted. SPCR2 = 0031h
; The transmitter is taken out of reset.
```

15.11 Data Packing Examples

This section shows two ways to implement data packing in the McBSP.

15.11.1 Data Packing Using Frame Length and Word Length

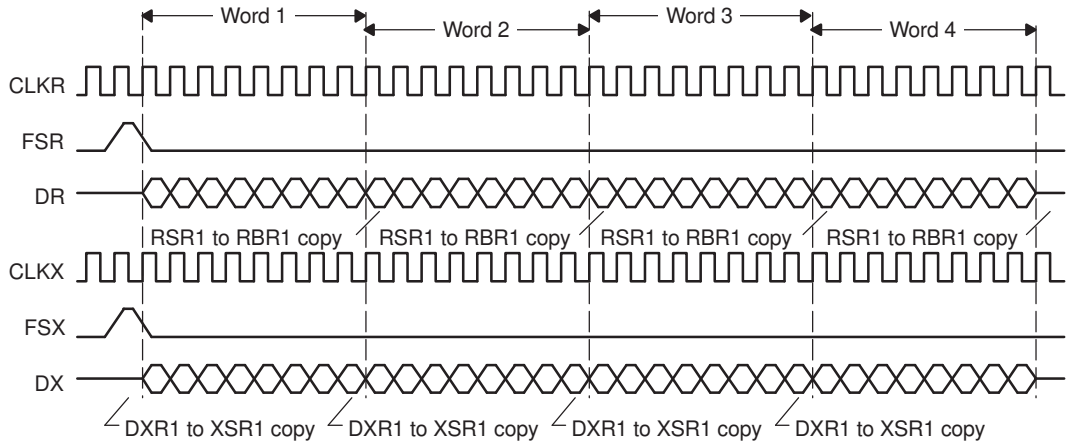
Frame length and word length can be manipulated to effectively pack data. For example, consider a situation where four 8-bit words are transferred in a single-phase frame as shown in Figure 15-61. In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000011b: 4-word frame
- (R/X)WDLEN1 = 000b: 8-bit words

Four 8-bit data words are transferred to and from the McBSP by the CPU or by the DMA controller. Thus, four reads from DRR1 and four writes to DXR1 are necessary for each frame.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-61. Four 8-Bit Data Words Transferred To/From the McBSP



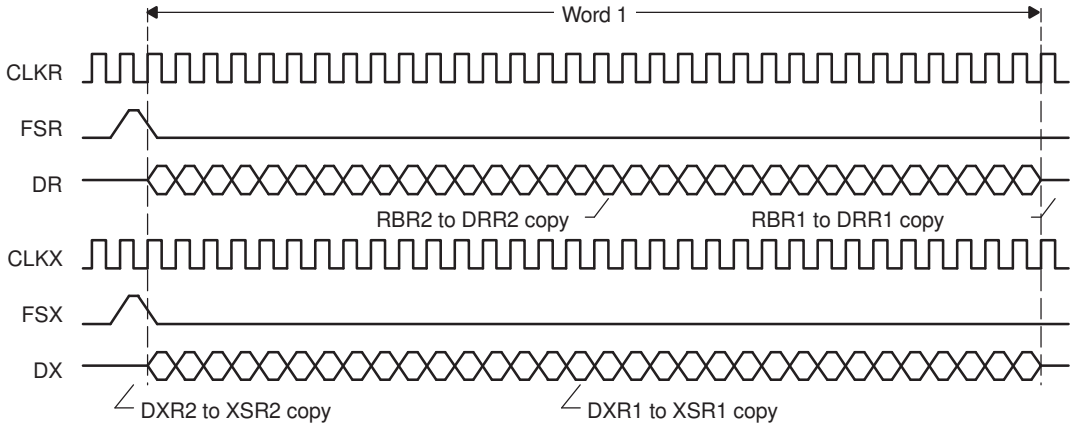
This data can also be treated as a single-phase frame consisting of one 32-bit data word, as shown in Figure 15-62. In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000000b: 1-word frame
- (R/X)WDLEN1 = 101b: 32-bit word

Two 16-bit data words are transferred to and from the McBSP by the CPU or DMA controller. Thus, two reads, from DRR2 and DRR1, and two writes, to DXR2 and DXR1, are necessary for each frame. This results in only half the number of transfers compared to the previous case. This manipulation reduces the percentage of bus time required for serial port data movement.

NOTE: When the word length is larger than 16 bits, make sure you access DRR2/DXR2 before you access DRR1/DXR1. McBSP activity is tied to accesses of DRR1/DXR1. During the reception of 24-bit or 32-bit words, read DRR2 and then read DRR1. Otherwise, the next RBR[1,2]-to-DRR[1,2] copy occurs before DRR2 is read. Similarly, during the transmission of 24-bit or 32-bit words, write to DXR2 and then write to DXR1. Otherwise, the next DXR[1,2]-to-XSR[1,2] copy occurs before DXR2 is loaded with new data.

Figure 15-62. One 32-Bit Data Word Transferred To/From the McBSP



Copyright © 2012–2019, Texas Instruments Incorporated

15.11.2 Data Packing Using Word Length and the Frame-Synchronization Ignore Function

When there are multiple words per frame, you can implement data packing by increasing the word length (defining a serial word with more bits) and by ignoring frame-synchronization pulses. First, consider Figure 15-63, which shows the McBSP operating at the maximum packet frequency. Here, each frame only has a single 8-bit word. Notice the frame-synchronization pulse that initiates each frame transfer for reception and for transmission. For reception, this configuration requires one read operation for each word. For transmission, this configuration requires one write operation for each word.

Figure 15-63. 8-Bit Data Words Transferred at Maximum Packet Frequency

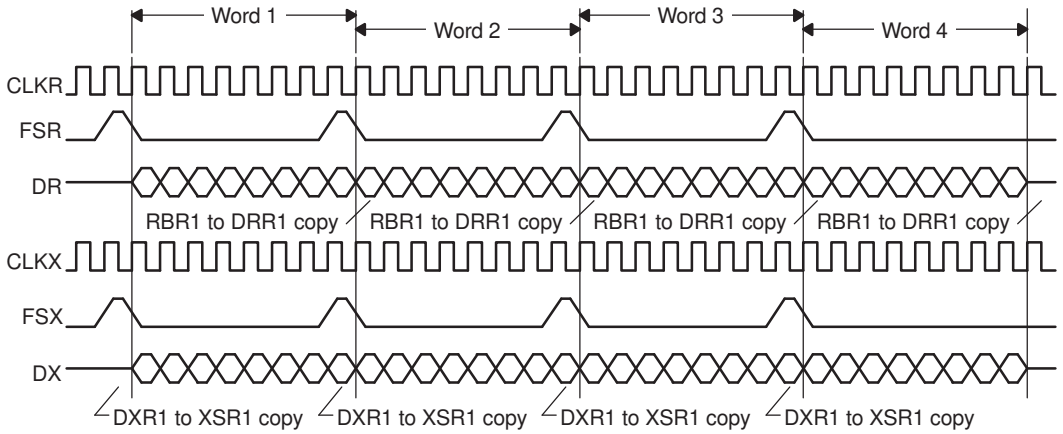
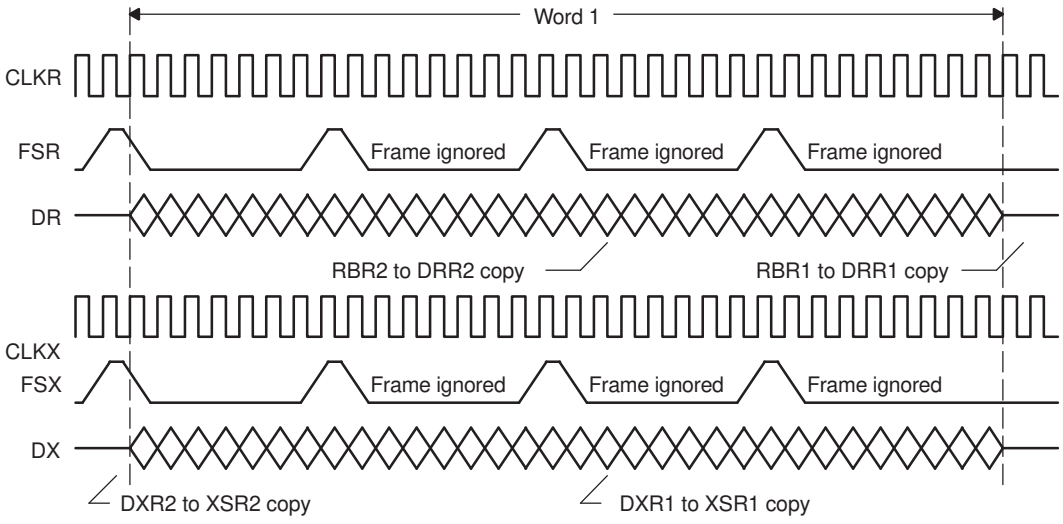


Figure 15-64 shows the McBSP configured to treat this data stream as a continuous 32-bit word. In this example, the McBSP responds to an initial frame-synchronization pulse. However, (R/X)FIG = 1 so that the McBSP ignores subsequent pulses. Only two read transfers or two write transfers are needed every 32 bits. This configuration effectively reduces the required bus bandwidth to half the bandwidth needed to transfer four 8-bit words.

Figure 15-64. Configuring the Data Stream of Figure 15-63 as a Continuous 32-Bit Word



15.12 McBSP Registers

This section describes the McBSP registers.

Table 15-71 shows the registers accessible on each McBSP. Section 15.12.2 through Section 15.12.11 describe the register bits.

Copyright © 2012–2019, Texas Instruments Incorporated

15.12.1 Register Summary

Table 15-71. McBSP Register Summary

Name	McBSP-A Address	Type	Reset Value	Description
Data Registers, Receive, Transmit				
DRR2	0x5000	R	0x0000	McBSP Data Receive Register 2
DRR1	0x5001	R	0x0000	McBSP Data Receive Register 1
DXR2	0x5002	W	0x0000	McBSP Data Transmit Register 2
DXR1	0x5003	W	0x0000	McBSP Data Transmit Register 1
McBSP Control Registers				
SPCR2	0x5004	R/W	0x0000	McBSP Serial Port Control Register 2
SPCR1	0x5005	R/W	0x0000	McBSP Serial Port Control Register 1
RCR2	0x5006	R/W	0x0000	McBSP Receive Control Register 2
RCR1	0x5007	R/W	0x0000	McBSP Receive Control Register 1
XCR2	0x5008	R/W	0x0000	McBSP Transmit Control Register 2
XCR1	0x5009	R/W	0x0000	McBSP Transmit Control Register 1
SRGR2	0x500A	R/W	0x0000	McBSP Sample Rate Generator Register 2
SRGR1	0x500B	R/W	0x0000	McBSP Sample Rate Generator Register 1
Multichannel Control Registers				
MCR2	0x500C	R/W	0x0000	McBSP Multichannel Register 2
MCR1	0x500D	R/W	0x0000	McBSP Multichannel Register 1
RCERA	0x500E	R/W	0x0000	McBSP Receive Channel Enable Register Partition A
RCERB	0x500F	R/W	0x0000	McBSP Receive Channel Enable Register Partition B
XCERA	0x5010	R/W	0x0000	McBSP Transmit Channel Enable Register Partition A
XCERB	0x5011	R/W	0x0000	McBSP Transmit Channel Enable Register Partition B
PCR	0x5012	R/W	0x0000	McBSP Pin Control Register
RCERC	0x5013	R/W	0x0000	McBSP Receive Channel Enable Register Partition C
RCERD	0x5014	R/W	0x0000	McBSP Receive Channel Enable Register Partition D
XCERC	0x5015	R/W	0x0000	McBSP Transmit Channel Enable Register Partition C
XCERD	0x5016	R/W	0x0000	McBSP Transmit Channel Enable Register Partition D
RCERE	0x5017	R/W	0x0000	McBSP Receive Channel Enable Register Partition E
RCERF	0x5018	R/W	0x0000	McBSP Receive Channel Enable Register Partition F
XCERE	0x5019	R/W	0x0000	McBSP Transmit Channel Enable Register Partition E
XCERF	0x501A	R/W	0x0000	McBSP Transmit Channel Enable Register Partition F
RCERG	0x501B	R/W	0x0000	McBSP Receive Channel Enable Register Partition G
RCERH	0x501C	R/W	0x0000	McBSP Receive Channel Enable Register Partition H
XCERG	0x501D	R/W	0x0000	McBSP Transmit Channel Enable Register Partition G
XCERH	0x501E	R/W	0x0000	McBSP Transmit Channel Enable Register Partition H
MFFINT	0x5023	R/W	0x0000	McBSP Interrupt Enable Register

15.12.2 Data Receive Registers (DRR[1,2])

The CPU or the DMA controller reads received data from one or both of the data receive registers (see Figure 15-65). If the serial word length is 16 bits or smaller, only DRR1 is used. If the serial length is larger than 16 bits, both DRR1 and DRR2 are used and DRR2 holds the most significant bits. Each frame of receive data in the McBSP can have one phase or two phases, each with its own serial word length.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-65. Data Receive Registers (DRR2 and DRR1)

DDR2

15	0
High part of receive data (for 20-, 24- or 32-bit data)	
R/W-0	

DDR1

15	0
Receive data (for 8-, 12-, or 16-bit data) or low part of receive data (for 20-, 24- or 32-bit data)	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

15.12.2.1 Data Travel From Data Receive Pins to the Registers

If the serial word length is 16 bits or smaller, receive data on the MDRx pin is shifted into receive shift register 1 (RSR1) and then copied into receive buffer register 1 (RBR1). The content of RBR1 is then copied to DRR1, which can be read by the CPU or by the DMA controller. The RSRs and RBRs are not accessible to the user.

If the serial word length is larger than 16 bits, receive data on the MDRx pin is shifted into both of the receive shift registers (RSR2, RSR1) and then copied into both of the receive buffer registers (RBR2, RBR1). The content of the RBRs is then copied into both of the DRRs, which can be read by the CPU or by the DMA controller.

If companding is used during the copy from RBR1 to DRR1 (RCOMPAND = 10b or 11b), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DRR1. If companding is disabled, the data copied from RBR[1,2] to DRR[1,2] is justified and bit filled according to the RJUST bits.

15.12.3 Data Transmit Registers (DXR[1,2])

For transmission, the CPU or the DMA controller writes data to one or both of the data transmit registers (see Figure 15-66). If the serial word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, both DXR1 and DXR2 are used and DXR2 holds the most significant bits. Each frame of transmit data in the McBSP can have one phase or two phases, each with its own serial word length.

Figure 15-66. Data Transmit Registers (DXR2 and DXR1)

DXR2

15	0
High part of transmit data (for 20-, 24- or 32-bit data)	
R/W-0	

DXR1

15	0
Transmit data (for 8-, 12-, or 16-bit data) or low part of receive data (for 20-, 24- or 32-bit data)	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

15.12.3.1 Data Travel From Registers to Data Transmit (DX) Pins

If the serial word length is 16 bits or fewer, data written to DXR1 is copied to transmit shift register 1 (XSR1). From XSR1, the data is shifted onto the DX pin one bit at a time. The XSRs are not accessible.

If the serial word length is more than 16 bits, data written to DXR1 and DXR2 is copied to both transmit shift registers (XSR2, XSR1). From the XSRs, the data is shifted onto the DX pin one bit at a time.

If companding is used during the transfer from DXR1 to XSR1 (XCOMPAND = 10b or 11b), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in the μ-law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXR(s) to the XSR(s) without modification.

Copyright © 2012–2019, Texas Instruments Incorporated

15.12.4 Serial Port Control Registers (SPCR[1,2])

Each McBSP has two serial port control registers, SPCR1 (Table 15-72) and SPCR2 (Table 15-73). These registers enable you to:

- Control various McBSP modes: digital loopback mode (DLB), sign-extension and justification mode for reception (RJUST), clock stop mode (CLKSTP), interrupt modes (RINTM and XINTM), emulation mode (FREE and SOFT)
- Turn on and off the DX-pin delay enabler (DXENA)
- Check the status of receive and transmit operations (RSYNCERR, XSYNCERR, RFULL, XEMPTY, RRDY, XRDY)
- Reset portions of the McBSP (RRST, XRST, FRST, GRST)

15.12.4.1 Serial Port Control 1 Register (SPCR1)

The serial port control 1 register (SPCR1) is shown in Figure 15-67 and described in Table 15-72.

Figure 15-67. Serial Port Control 1 Register (SPCR1)

15	14	13	12	11	10	8
DLB	RJUST		CLKSTP		Reserved	
R/W-0	R/W-0		R/W-0		R-0	

7	6	5	4	3	2	1	0
DXENA	Reserved	RINTM		RSYNCERR	RFULL	RRDY	RRST
R/W-0	R/W-0	R/W-0		R/W-0	R-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-72. Serial Port Control 1 Register (SPCR1) Field Descriptions

Bit	Field	Value	Description
15	DLB		Digital loopback mode bit. DLB disables or enables the digital loopback mode of the McBSP:
		0	Disabled
			Internal DR is supplied by the MDRx pin. Internal FSR and internal MCLKR can be supplied by their respective pins or by the sample rate generator, depending on the mode bits FSRM and CLKRM.
			Internal DX is supplied by the MDXx pin. Internal FSX and internal CLKX are supplied by their respective pins or are generated internally, depending on the mode bits FSXM and CLKXM.
		1	Enabled
			Internal receive signals are supplied by internal transmit signals:
			MDRx connected to MDXx
			MFSRx connected to MFSXx
			MCLKR connected to MCLKXx
			This mode allows you to test serial port code with a single DSP. The McBSP transmitter directly supplies data, frame synchronization, and clocking to the McBSP receiver.
14-13	RJUST	0-3h	Receive sign-extension and justification mode bits. During reception, RJUST determines how data is justified and bit filled before being passed to the data receive registers (DRR1, DRR2).
			RJUST is ignored if you enable a companding mode with the RCOMPAND bits. In a companding mode, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1.
			For more details about the effects of RJUST, see Section 15.8.13, *Set the Receive Sign-Extension and Justification Mode*
		0	Right justify the data and zero fill the MSBs.
		1h	Right justify the data and sign-extend the data into the MSBs.
		2h	Left justify the data and zero fill the LSBs.
		3h	Reserved (do not use)

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-72. Serial Port Control 1 Register (SPCR1) Field Descriptions (continued)

Bit	Field	Value	Description
12-11	CLKSTP	0-3h	Clock stop mode bits. CLKSTP allows you to use the clock stop mode to support the SPI master-slave protocol. If you will not be using the SPI protocol, you can clear CLKSTP to disable the clock stop mode. In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). For more details, see Section 15.8.5, *Enable/Disable the Clock Stop*.
		0-1h	Clock stop mode is disabled.
		2h	Clock stop mode, without clock delay
		3h	Clock stop mode, with half-cycle clock delay
10-8	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
7	DXENA		DX delay enabler mode bit. DXENA controls the delay enabler for the DX pin. The enabler creates an extra delay for turn-on time (for the length of the delay, see the device-specific data sheet). For more details about the effects of DXENA, see Section 15.9.13, *Set the Transmit DXENA Mode*.
		0	DX delay enabler off
		1	DX delay enabler on
6	Reserved	0	Reserved
5-4	RINTM	0-3h	Receive interrupt mode bits. RINTM determines which event in the McBSP receiver generates a receive interrupt (RINT) request. If RINT is properly enabled inside the CPU, the CPU services the interrupt request; otherwise, the CPU ignores the request.
		0	The McBSP sends a receive interrupt (RINT) request to the CPU when the RRDY bit changes from 0 to 1, indicating that receive data is ready to be read (the content of RBR[1,2] has been copied to DRR[1,2]): Regardless of the value of RINTM, you can check RRDY to determine whether a word transfer is complete. The McBSP sends a RINT request to the CPU when 16 enabled bits have been received on the DR pin.
		1h	In the multichannel selection mode, the McBSP sends a RINT request to the CPU after every 16-channel block is received in a frame. Outside of the multichannel selection mode, no interrupt request is sent.
		2h	The McBSP sends a RINT request to the CPU when each receive frame-synchronization pulse is detected. The interrupt request is sent even if the receiver is in its reset state.
		3h	The McBSP sends a RINT request to the CPU when the RSYNCERR bit is set, indicating a receive frame-synchronization error. Regardless of the value of RINTM, you can check RSYNCERR to determine whether a receive frame-synchronization error occurred.
3	RSYNCERR		Receive frame-sync error bit. RSYNCERR is set when a receive frame-sync error is detected by the McBSP. If RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU when RSYNCERR is set. The flag remains set until you write a 0 to it or reset the receiver.
		0	No error
		1	Receive frame-synchronization error. For more details about this error, see Section 15.5.3, *Unexpected Receive Frame-Synchronization Pulse*.
2	RFULL		Receiver full bit. RFULL is set when the receiver is full with new data and the previously received data has not been read (receiver-full condition). For more details about this condition, see Section 15.5.2, *Overrun in the Receiver*.
		0	No receiver-full condition
		1	Receiver-full condition: RSR[1,2] and RBR[1,2] are full with new data, but the previous data in DRR[1,2] has not been read.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-72. Serial Port Control 1 Register (SPCR1) Field Descriptions (continued)

Bit	Field	Value	Description
1	RRDY		Receiver ready bit. RRDY is set when data is ready to be read from DRR[1,2]. Specifically, RRDY is set in response to a copy from RBR1 to DRR1. If the receive interrupt mode is RINTM = 00b, the McBSP sends a receive interrupt request to the CPU when RRDY changes from 0 to 1. Also, when RRDY changes from 0 to 1, the McBSP sends a receive synchronization event (REVT) signal to the DMA controller.
		0	Receiver not ready When the content of DRR1 is read, RRDY is automatically cleared.
		1	Receiver ready: New data can be read from DRR[1,2]. Important: If both DRRs are required (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.
0	RRST		Receiver reset bit. You can use RRST to take the McBSP receiver into and out of its reset state. This bit has a negative polarity; RRST = 0 indicates the reset state. To read about the effects of a receiver reset, see Section 15.10.2, *Resetting and Initializing a McBSP*.
		0	If you read a 0, the receiver is in its reset state. If you write a 0, you reset the receiver.
		1	If you read a 1, the receiver is enabled. If you write a 1, you enable the receiver by taking it out of its reset state.

Copyright © 2012–2019, Texas Instruments Incorporated

15.12.4.2 Serial Port Control 2 Register (SPCR2)

The serial port control 2 register (SPCR2) is shown in Figure 15-68 and described in Table 15-73.

Figure 15-68. Serial Port Control 2 Register (SPCR2)

15					10	9	8
Reserved						FREE	SOFT
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
FRST	GRST	XINTM		XSYNCERR	XEMPTY	XRDY	XRST
R/W-0	R/W-0	R/W-0		R/W-0	R-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-73. Serial Port Control 2 Register (SPCR2) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
9	FREE		Free run bit. When a breakpoint is encountered in the high-level language debugger, FREE determines whether the McBSP transmit and receive clocks continue to run or whether they are affected as determined by the SOFT bit. When one of the clocks stops, the corresponding data transfer (transmission or reception) stops.
8	SOFT		Soft stop bit. When FREE = 0, SOFT determines the response of the McBSP transmit and receive clocks when a breakpoint is encountered in the high-level language debugger. When one of the clocks stops, the corresponding data transfer (transmission or reception) stops.
7	FRST		Frame-synchronization logic reset bit. The sample rate generator of the McBSP includes frame-synchronization logic to generate an internal frame-synchronization signal. You can use FRST to take the frame-synchronization logic into and out of its reset state. This bit has a negative polarity; FRST = 0 indicates the reset state.
		0	If you read a 0, the frame-synchronization logic is in its reset state. If you write a 0, you reset the frame-synchronization logic. In the reset state, the frame-synchronization logic does not generate a frame-synchronization signal (FSG).
		1	If you read a 1, the frame-synchronization logic is enabled. If you write a 1, you enable the frame-synchronization logic by taking it out of its reset state. When the frame-synchronization logic is enabled (FRST = 1) and the sample rate generator as a whole is enabled (GRST = 1), the frame-synchronization logic generates the frame-synchronization signal FSG as programmed.
6	GRST		Sample rate generator reset bit. You can use GRST to take the McBSP sample rate generator into and out of its reset state. This bit has a negative polarity; GRST = 0 indicates the reset state. To read about the effects of a sample rate generator reset, see Section 15.10.2, *Resetting and Initializing a McBSP*.
		0	If you read a 0, the sample rate generator is in its reset state. If you write a 0, you reset the sample rate generator. If GRST = 0 due to a reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 due to program code, CLKG and FSG are both driven low (inactive).
		1	If you read a 1, the sample rate generator is enabled. If you write a 1, you enable the sample rate generator by taking it out of its reset state. When enabled, the sample rate generator generates the clock signal CLKG as programmed in the sample rate generator registers. If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-73. Serial Port Control 2 Register (SPCR2) Field Descriptions (continued)

Bit	Field	Value	Description
5-4	XINTM	0-3h	Transmit interrupt mode bits. XINTM determines which event in the McBSP transmitter generates a transmit interrupt (XINT) request. If XINT is properly enabled, the CPU services the interrupt request; otherwise, the CPU ignores the request.
		0	The McBSP sends a transmit interrupt (XINT) request to the CPU when the XRDY bit changes from 0 to 1, indicating that transmitter is ready to accept new data (the content of DXR[1,2] has been copied to XSR[1,2]). Regardless of the value of XINTM, you can check XRDY to determine whether a word transfer is complete.
			The McBSP sends an XINT request to the CPU when 16 enabled bits have been transmitted on the DX pin.
		1h	In the multichannel selection mode, the McBSP sends an XINT request to the CPU after every 16-channel block is transmitted in a frame. Outside of the multichannel selection mode, no interrupt request is sent.
		2h	The McBSP sends an XINT request to the CPU when each transmit frame-synchronization pulse is detected. The interrupt request is sent even if the transmitter is in its reset state.
		3h	The McBSP sends an XINT request to the CPU when the XSYNCERR bit is set, indicating a transmit frame-synchronization error. Regardless of the value of XINTM, you can check XSYNCERR to determine whether a transmit frame-synchronization error occurred.
3	XSYNCERR		Transmit frame-synchronization error bit. XSYNCERR is set when a transmit frame-synchronization error is detected by the McBSP. If XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU when XSYNCERR is set. The flag remains set until you write a 0 to it or reset the transmitter. If XINTM = 11b, writing a 1 to XSYNCERR triggers a transmit interrupt just as if a transmit frame-synchronization error occurred. For details about this error see Section 15.5.5, *Unexpected Transmit Frame-Synchronization Pulse*.
		0	No error
		1	Transmit frame-synchronization error
2	XEMPTY		Transmitter empty bit. XEMPTY is cleared when the transmitter is ready to send new data but no new data is available (transmitter-empty condition). This bit has a negative polarity; a transmitter-empty condition is indicated by XEMPTY = 0.
		0	Transmitter-empty condition Typically this indicates that all the bits of the current word have been transmitted but there is no new data in DXR1. XEMPTY is also cleared if the transmitter is reset and then restarted. For more details about this error condition, see Section 15.5.4.3, *Underflow in the Transmitter*.
		1	No transmitter-empty condition
1	XRDY		Transmitter ready bit. XRDY is set when the transmitter is ready to accept new data in DXR[1,2]. Specifically, XRDY is set in response to a copy from DXR1 to XSR1. If the transmit interrupt mode is XINTM = 00b, the McBSP sends a transmit interrupt (XINT) request to the CPU when XRDY changes from 0 to 1. Also, when XRDY changes from 0 to 1, the McBSP sends a transmit synchronization event (XEVT) signal to the DMA controller.
		0	Transmitter not ready When DXR1 is loaded, XRDY is automatically cleared.
		1	Transmitter ready: DXR[1,2] is ready to accept new data. If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-73. Serial Port Control 2 Register (SPCR2) Field Descriptions (continued)

Bit	Field	Value	Description
0	XRST		Transmitter reset bit. You can use XRST to take the McBSP transmitter into and out of its reset state. This bit has a negative polarity; XRST = 0 indicates the reset state. To read about the effects of a transmitter reset, see Section 15.10.2, *Resetting and Initializing a McBSP*.
		0	If you read a 0, the transmitter is in its reset state. If you write a 0, you reset the transmitter.
		1	If you read a 1, the transmitter is enabled. If you write a 1, you enable the transmitter by taking it out of its reset state.

15.12.5 Receive Control Registers (RCR[1, 2])

Each McBSP has two receive control registers, RCR1 (Table 15-74) and RCR2 (Table 15-76). These registers enable you to:

- Specify one or two phases for each frame of receive data (RPHASE)
- Define two parameters for phase 1 and (if necessary) phase 2: the serial word length (RWDLEN1, RWDLEN2) and the number of words (RFRLEN1, RFRLEN2)
- Choose a receive companding mode, if any (RCOMPAND)
- Enable or disable the receive frame-synchronization ignore function (RFIG)
- Choose a receive data delay (RDATDLY)

15.12.5.1 Receive Control Register 1 (RCR1)

The receive control register 1 (RCR1) is shown in Figure 15-69 and described in Table 15-74.

Figure 15-69. Receive Control Register 1 (RCR1)

15	14	8
Reserved	RFRLEN1	
R-0	R/W-0	

7	5	4	0
RWDLEN1		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-74. Receive Control Register 1 (RCR1) Field Descriptions

Bit	Field	Value	Description
15	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
14-8	RFRLEN1	0-7Fh	Receive frame length 1 (1 to 128 words). Each frame of receive data can have one or two phases, depending on value that you load into the RPHASE bit. If a single-phase frame is selected, RFRLEN1 in RCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, RFRLEN1 determines the number of serial words in phase 1 of the frame, and RFRLEN2 in RCR2 determines the number of words in phase 2 of the frame. The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 15-75 for a summary of how you determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the RFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into RFRLEN1.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-74. Receive Control Register 1 (RCR1) Field Descriptions (continued)

Bit	Field	Value	Description
7-5	RWDLEN1	0-7h	Receive word length 1. Each frame of receive data can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 in RCR1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 in RCR2 determines the word length in phase 2 of the frame.
		0	8 bits
		1h	12 bits
		2h	16 bits
		3h	20 bits
		4h	24 bits
		5h	32 bits
		6h-7h	Reserved (do not use)
4-0	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.

Table 15-75. Frame Length Formula for Receive Control 1 Register (RCR1)

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 ≤ RFRLEN1 ≤ 127	Not used	(RFRLEN1 + 1) words
1	0 ≤ RFRLEN1 ≤ 127	0 ≤ RFRLEN2 ≤ 127	(RFRLEN1 + 1) + (RFRLEN2 + 1) words

15.12.5.2 Receive Control Register 2 (RCR2)

The receive control register 2 (RCR2) is shown in Figure 15-70 and described in Table 15-76.

Figure 15-70. Receive Control Register 2 (RCR2)

15	14–8
RPHASE	RFRLEN2
R/W-0	R/W-0

7–5	4–3	2	1–0
RWDLEN2	RCOMPAND	RFIG	RDATDLY
R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-76. Receive Control Register 2 (RCR2) Field Descriptions

Bit	Field	Value	Description
15	RPHASE		Receive phase number bit. RPHASE determines whether the receive frame has one phase or two phases. For each phase you can define the serial word length and the number of serial words in the phase. To set up phase 1, program RWDLEN1 (word length) and RFRLEN1 (number of words). To set up phase 2 (if there are two phases), program RWDLEN2 and RFRLEN2.
		0	Single-phase frame
			The receive frame has only one phase, phase 1.
		1	Dual-phase frame
			The receive frame has two phases, phase 1 and phase 2.
14-8		0-7Fh	Receive frame length 2 (1 to 128 words). Each frame of receive data can have one or two phases, depending on value that you load into the RPHASE bit. If a single-phase frame is selected, RFRLEN1 in RCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, RFRLEN1 determines the number of serial words in phase 1 of the frame, and RFRLEN2 in RCR2 determines the number of words in phase 2 of the frame. The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 15-77 for a summary of how to determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period.
			Program the RFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For example, if you want a phase length of 128 words in phase 2, load 127 into RFRLEN2.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-76. Receive Control Register 2 (RCR2) Field Descriptions (continued)

Bit	Field	Value	Description
7-5	RWDLEN2	0-7h	Receive word length 2. Each frame of receive data can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 in RCR1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 in RCR2 determines the word length in phase 2 of the frame.
		0	8 bits
		1h	12 bits
		2h	16 bits
		3h	20 bits
		4h	24 bits
		5h	32 bits
		6h-7h	Reserved (do not use)
4-3	RCOMPAND	0-3h	Receive companding mode bits. Companding (COMpress and exPAND) hardware allows compression and expansion of data in either μ-law or A-law format. RCOMPAND allows you to choose one of the following companding modes for the McBSP receiver: For more details about these companding modes, see Section 15.1.5, *Companding (Compressing and Expanding) Data*.
		0	No companding, any size data, MSB received first
		1h	No companding, 8-bit data, LSB received first
		2h	μ-law companding, 8-bit data, MSB received first
		3h	A-law companding, 8-bit data, MSB received first
2	RFIG		Receive frame-synchronization ignore bit. If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse. For more details about the frame-synchronization error condition, see Section 15.5.3, *Unexpected Receive Frame-Synchronization Pulse*. Setting RFIG causes the serial port to ignore unexpected frame-synchronization signals during reception. For more details on the effects of RFIG, see Section 15.8.10.1, *Enable/Disable the Receive Frame-Synchronization Ignore Function*.
		0	Frame-synchronization detect. An unexpected FSR pulse causes the receiver to discard the contents of RSR[1,2] in favor of the new incoming data. The receiver: 1. Aborts the current data transfer 2. Sets RSYNCERR in SPCR1 3. Begins the transfer of a new data word
		1	Frame-synchronization ignore. An unexpected FSR pulse is ignored. Reception continues uninterrupted.
1-0	RDATDLY	0-3h	Receive data delay bits. RDATDLY specifies a data delay of 0, 1, or 2 receive clock cycles after frame-synchronization and before the reception of the first bit of the frame. For more details, see Section 15.8.12, *Set the Receive Data Delay*.
		0	0-bit data delay
		1h	1-bit data delay
		2h	2-bit data delay
		3h	Reserved (do not use)

Table 15-77. Frame Length Formula for Receive Control 2 Register (RCR2)

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 ≤ RFRLEN1 ≤ 127	Not used	(RFRLEN1 + 1) words
1	0 ≤ RFRLEN1 ≤ 127	0 ≤ RFRLEN2 ≤ 127	(RFRLEN1 + 1) + (RFRLEN2 + 1) words

15.12.6 Transmit Control Registers (XCR1 and XCR2)

Each McBSP has two transmit control registers, XCR1 (Table 15-78) and XCR2 (Table 15-80). These registers enable you to:

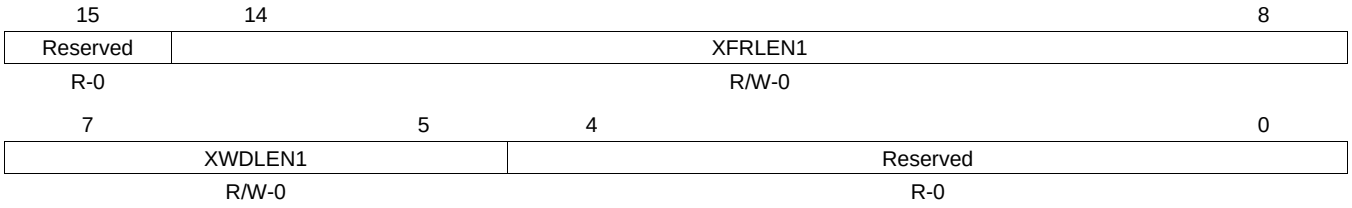
Copyright © 2012–2019, Texas Instruments Incorporated

- Specify one or two phases for each frame of transmit data (XPHASE)
- Define two parameters for phase 1 and (if necessary) phase 2: the serial word length (XWDLEN1, XWDLEN2) and the number of words (XFRLEN1, XFRLEN2)
- Choose a transmit companding mode, if any (XCOMPAND)
- Enable or disable the transmit frame-sync ignore function (XFIG)
- Choose a transmit data delay (XDATDLY)

15.12.6.1 Transmit Control 1 Register (XCR1)

The transmit control 1 register (XCR1) is shown in Figure 15-71 and described in Table 15-78.

Figure 15-71. Transmit Control 1 Register (XCR1)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-78. Transmit Control 1 Register (XCR1) Field Descriptions

Bit	Field	Value	Description
15	Reserved	0	Reserved bit. Read-only; returns 0 when read.
14-8	XFRLEN1	0-7Fh	Transmit frame length 1 (1 to 128 words). Each frame of transmit data can have one or two phases, depending on value that you load into the XPHASE bit. If a single-phase frame is selected, XFRLEN1 in XCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, XFRLEN1 determines the number of serial words in phase 1 of the frame and XFRLEN2 in XCR2 determines the number of words in phase 2 of the frame. The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 15-79 for a summary of how you determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the XFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.
7-5	XWDLEN1	0-3h	Transmit word length 1. Each frame of transmit data can have one or two phases, depending on the value that you load into the XPHASE bit. If a single-phase frame is selected, XWDLEN1 in XCR1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame and XWDLEN2 in XCR2 determines the word length in phase 2 of the frame.
		0	8 bits
		1h	12 bits
		2h	16 bits
		3h	20 bits
		4h	24 bits
		5h	32 bits
		6h-7h	Reserved (do not use)
4-0	Reserved	0	Reserved bits. They are read-only bits and return 0s when read.

Table 15-79. Frame Length Formula for Transmit Control 1 Register (XCR1)

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 ≤ XFRLEN1 ≤ 127	Not used	(XFRLEN1 + 1) words
1	0 ≤ XFRLEN1 ≤ 127	0 ≤ XFRLEN2 ≤ 127	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

Copyright © 2012–2019, Texas Instruments Incorporated

15.12.6.2 Transmit Control 2 Register (XCR2)

The transmit control 2 register (XCR2) is shown in Figure 15-72 and described in Table 15-80.

Figure 15-72. Transmit Control 2 Register (XCR2)

15	14	8
XPHASE	XFRLEN2	
R/W-0	R/W-0	

7	5	4	3	2	1	0
XWDLEN2		XCOMPAND		XFIG	XDATDLY	
R/W-0		R/W-0		R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 15-80. Transmit Control 2 Register (XCR2) Field Descriptions

Bit	Field	Value	Description
15	XPHASE		Transmit phase number bit. XPHASE determines whether the transmit frame has one phase or two phases. For each phase you can define the serial word length and the number of serial words in the phase. To set up phase 1, program XWDLEN1 (word length) and XFRLEN1 (number of words). To set up phase 2 (if there are two phases), program XWDLEN2 and XFRLEN2.
		0	Single-phase frame
			The transmit frame has only one phase, phase 1.
		1	Dual-phase frame
			The transmit frame has two phases, phase 1 and phase 2.
14-8	XFRLEN2	0-7Fh	Transmit frame length 2 (1 to 128 words). Each frame of transmit data can have one or two phases, depending on value that you load into the XPHASE bit. If a single-phase frame is selected, XFRLEN1 in XCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, XFRLEN1 determines the number of serial words in phase 1 of the frame and XFRLEN2 in XCR2 determines the number of words in phase 2 of the frame. The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 15-81 for a summary of how to determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period.
			Program the XFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.
7-5	XWDLEN2	0-7h	Transmit word length 2. Each frame of transmit data can have one or two phases, depending on the value that you load into the XPHASE bit. If a single-phase frame is selected, XWDLEN1 in XCR1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame and XWDLEN2 in XCR2 determines the word length in phase 2 of the frame.
		0	8 bits
		1h	12 bits
		2h	16 bits
		3h	20 bits
		4h	24 bits
		5h	32 bits
		6h-7h	Reserved (do not use)
4-3	XCOMPAND	0-3h	Transmit companding mode bits. Companding (COMpress and exPAND) hardware allows compression and expansion of data in either μ-law or A-law format. For more details, see Section 15.1.5.
			XCOMPAND allows you to choose one of the following companding modes for the McBSP transmitter. For more details about these companding modes, see Section 15.1.5 .
		0	No companding, any size data, MSB transmitted first
		1h	No companding, 8-bit data, LSB transmitted first
		2h	μ-law companding, 8-bit data, MSB transmitted first
		3h	A-law companding, 8-bit data, MSB transmitted first

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-80. Transmit Control 2 Register (XCR2) Field Descriptions (continued)

Bit	Field	Value	Description
2	XFIG		Transmit frame-synchronization ignore bit. If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse. For more details about the frame-synchronization error condition, see Section 15.5.5. Setting XFIG causes the serial port to ignore unexpected frame-synchronization pulses during transmission. For more details on the effects of XFIG, see Section 15.9.10.
		0	Frame-synchronization detect. An unexpected FSX pulse causes the transmitter to discard the content of XSR[1,2]. The transmitter: 1. Aborts the present transmission 2. Sets XSYNCERR in SPCR2 3. Begins a new transmission from DXR[1,2]. If new data was written to DXR[1,2] since the last DXR[1,2]-to-XSR[1,2] copy, the current value in XSR[1,2] is lost. Otherwise, the same data is transmitted.
		1	Frame-synchronization ignore. An unexpected FSX pulse is ignored. Transmission continues uninterrupted.
1-0	XDATDLY	0-3h	Transmit data delay bits. XDATDLY specifies a data delay of 0, 1, or 2 transmit clock cycles after frame synchronization and before the transmission of the first bit of the frame. For more details, see Section 15.9.12.
		0	0-bit data delay
		1h	1-bit data delay
		2h	2-bit data delay
		3h	Reserved (do not use)

Table 15-81. Frame Length Formula for Transmit Control 2 Register (XCR2)

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 ≤ XFRLEN1 ≤ 127	Not used	(XFRLEN1 + 1) words
1	0 ≤ XFRLEN1 ≤ 127	0 ≤ XFRLEN2 ≤ 127	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

15.12.7 Sample Rate Generator Registers (SRGR1 and SRGR2)

Each McBSP has two sample rate generator registers, SRGR1 (Table 15-82) and SRGR2 (Table 15-83). The sample rate generator can generate a clock signal (CLKG) and a frame-synchronization signal (FSG). The registers, SRGR1 and SRGR2, enable you to:

- Select the input clock source for the sample rate generator (CLKSM, in conjunction with the SCLKME bit of PCR)
- Divide down the frequency of CLKG (CLKGDV)
- Select whether internally-generated transmit frame-synchronization pulses are driven by FSG or by activity in the transmitter (FSGM).
- Specify the width of frame-synchronization pulses on FSG (FWID) and specify the period between those pulses (FPER)

When an external source (via the MCLKR or MCLKX pin) provides the input clock source for the sample rate generator:

- If the CLKX/MCLKR pin is used, the polarity of the input clock is selected with CLKXP/CLKRP of PCR.
- The GSYNC bit of SRGR2 allows you to make CLKG synchronized to an external frame-synchronization signal on the FSR pin, so that CLKG is kept in phase with the input clock.

15.12.7.1 Sample Rate Generator 1 Register (SRGR1)

The sample rate generator 1 register is shown in Figure 15-73 and described in Table 15-82.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 15-73. Sample Rate Generator 1 Register (SRGR1)

15							8
FWID							
R/W-0							
7							0
CLKGDV							
R/W-1							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-82. Sample Rate Generator 1 Register (SRGR1) Field Descriptions

Bit	Field	Value	Description
15-8	FWID	0-FFh	Frame-synchronization pulse width bits for FSG The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. For frame-synchronization pulses on FSG, (FWID + 1) is the pulse width in CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles: 0 ≤ FWID ≤ 255 1 ≤ (FWID + 1) ≤ 256 CLKG cycles The period between the frame-synchronization pulses on FSG is defined by the FPER bits.
7-0	CLKGDV	0-FFh	Divide-down value for CLKG. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is: CLKG frequency = (Input clock frequency)/ (CLKGDV + 1) The input clock is selected by the SCLKME and CLKSM bits: **SCLKME** / **CLKSM** / **Input Clock For Sample Rate Generator** 0 / 0 / Reserved 0 / 1 / LSPCLK 1 / 0 / Signal on MCLKR pin 1 / 1 / Signal on MCLKX pin

15.12.7.2 Sample Rate Generator 2 Register (SRGR2)

The sample rate generator 2 register (SRGR2) is shown in Figure 15-74 and described in Table 15-83.

Figure 15-74. Sample Rate Generator 2 Register (SRGR2)

15	14	13	12	11			8
GSYNC	Reserved	CLKSM	FSGM	FPER			
R/W-0	R/W-0	R/W-1	R/W-0	R/W-0			
7							0
FPER							
R/W-0							

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-83. Sample Rate Generator 2 Register (SRGR2) Field Descriptions

Bit	Field	Value	Description
15	GSYNC		Clock synchronization mode bit for CLKG. GSYNC is used only when the input clock source for the sample rate generator is external—on the MCLKR pin. When GSYNC = 1, the clock signal (CLKG) and the frame-synchronization signal (FSG) generated by the sample rate generator are made dependent on pulses on the FSR pin.
		0	No clock synchronization CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.
		1	Clock synchronization • CLKG is adjusted as necessary so that it is synchronized with the input clock on the MCLKR pin. • FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored. For more details, see Section 15.4.3, *Synchronizing Sample Rate Generator Outputs to an External Clock*.
14	Reserved		Reserved
13	CLKSM	0	Sample rate generator input clock mode bit. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is:) CLKG frequency = (input clock frequency)/ (CLKGDV + 1 CLKSM is used in conjunction with the SCLKME bit to determine the source for the input clock. A reset selects the CPU clock as the input clock and forces the CLKG frequency to ½ the LSPCLK frequency. The input clock for the sample rate generator is taken from the MCLKR pin, depending on the value of the SCLKME bit of PCR: **SCLKME** / **CLKSM** / **Input Clock For Sample Rate Generator** 0 / 0 / Reserved 1 / 0 / Signal on MCLKR pin
		1	The input clock for the sample rate generator is taken from the LSPCLK or from the MCLKX pin, depending on the value of the SCLKME bit of PCR: **SCLKME** / **CLKSM** / **Input Clock For Sample Rate Generator** 0 / 1 / LSPCLK 1 / 1 / Signal on MCLKX pin
12	FSGM		Sample rate generator transmit frame-synchronization mode bit. The transmitter can get frame synchronization from the FSX pin (FSXM = 0) or from inside the McBSP (FSXM = 1). When FSXM = 1, the FSGM bit determines how the McBSP supplies frame-synchronization pulses.
		0	If FSXM = 1, the McBSP generates a transmit frame-synchronization pulse when the content of DXR[1,2] is copied to XSR[1,2].
		1	If FSXM = 1, the transmitter uses frame-synchronization pulses generated by the sample rate generator. Program the FWID bits to set the width of each pulse. Program the FPER bits to set the period between pulses.
11-0	FPER	0-FFFh	Frame-synchronization period bits for FSG. The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. The period between frame-synchronization pulses on FSG is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles: 0 ≤ FPER ≤ 4095 1 ≤ (FPER + 1) ≤ 4096 CLKG cycles The width of each frame-synchronization pulse on FSG is defined by the FWID bits.

15.12.8 Multichannel Control Registers (MCR[1,2])

Each McBSP has two multichannel control registers. MCR1 (Table 15-84) has control and status bits (with an R prefix) for multichannel selection operation in the receiver. MCR2 (Table 15-85) contains the same type of bits (bit with an X prefix) for the transmitter. These registers enable you to:

Copyright © 2012–2019, Texas Instruments Incorporated

- Enable all channels or only selected channels for reception (RMCM)
- Choose which channels are enabled/disabled and masked/unmasked for transmission (XMCM)
- Specify whether two partitions (32 channels at a time) or eight partitions (128 channels at a time) can be used (RMCME for reception, XMCME for transmission)
- Assign blocks of 16 channels to partitions A and B when the 2-partition mode is selected (RPABLK and RPBBLK for reception, XPABLK and XPBBLK for transmission)
- Determine which block of 16 channels is currently involved in a data transfer (RCBLK for reception, XCBLK for transmission)

15.12.8.1 Multichannel Control 1 Register (MCR1)

The multichannel control 1 register (MCR1) is shown in Figure 15-75 and described in Table 15-84.

Figure 15-75. Multichannel Control 1 Register (MCR1)

15	10	9	8
Reserved		RMCME	RPBBLK
R-0		R/W-0	R/W-0

7	6	5	4	2	1	0
RPBBLK	RPABLK		RCBLK		Reserved	RMCM
R/W-0	R/W-0		R-0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-84. Multichannel Control 1 Register (MCR1) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
9	RMCME		Receive multichannel partition mode bit. RMCME is only applicable if channels can be individually enabled or disabled for reception (RMCM = 1). RMCME determines whether only 32 channels or all 128 channels are to be individually selectable.
		0	2-partition mode Only partitions A and B are used. You can control up to 32 channels in the receive multichannel selection mode (RMCM = 1). Assign 16 channels to partition A with the RPABLK bits. Assign 16 channels to partition B with the RPBBLK bits. You control the channels with the appropriate receive channel enable registers: RCERA: Channels in partition A RCERB: Channels in partition B
		1	8-partition mode All partitions (A through H) are used. You can control up to 128 channels in the receive multichannel selection mode. You control the channels with the appropriate receive channel enable registers: RCERA: Channels 0 through 15 RCERB: Channels 16 through 31 RCERC: Channels 32 through 47 RCERD: Channels 48 through 63 RCERE: Channels 64 through 79 RCERF: Channels 80 through 95 RCERG: Channels 96 through 111 RCERH: Channels 112 through 127

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-84. Multichannel Control 1 Register (MCR1) Field Descriptions (continued)

Bit	Field	Value	Description
8-7	RPBBLK	0-3h	Receive partition B block bits RPBBLK is only applicable if channels can be individually enabled or disabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver. The 128 receive channels of the McBSP are divided equally among 8 blocks (0 through 7). When RPBBLK is applicable, use RPBBLK to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B. Use the RPABLK bits to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A. If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the receiver is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The RCBLK bits are regularly updated to indicate which block is active. When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).
		0	Block 1: channels 16 through 31
		1h	Block 3: channels 48 through 63
		2h	Block 5: channels 80 through 95
		3h	Block 7: channels 112 through 127
6-5	RPABLK	0-3h	Receive partition A block bits RPABLK is only applicable if channels can be individually enabled or disabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver. See the description for RPBBLK (bits 8-7) for more information about assigning blocks to partitions A and B.
		0	Block 0: channels 0 through 15
		1h	Block 2: channels 32 through 47
		2h	Block 5: channels 64 through 79
		3h	Block 7: channels 96 through 111
4-2	RCBLK	0-7h	Receive current block indicator. RCBLK indicates which block fo 16 channels is involved in the current McBSP reception:
		0	Block 0: channels 0 through 15
		1h	Block 1: channels 16 through 31
		2h	Block 2: channels 32 through 47
		3h	Block 3: channels 48 through 63
		4h	Block 4: channels 64 through 79
		5h	Block 5: channels 80 through 95
		6h	Block 6: channels 96 through 111
		7h	Block 7: channels 112 through 127
1	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
0	RMCM		Receive multichannel selection mode bit. RMCM determines whether all channels or only selected channels are enabled for reception:
		0	All 128 channels are enabled.
		1	Multichanneled selection mode. Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit.

Copyright © 2012–2019, Texas Instruments Incorporated

15.12.8.2 Multichannel Control 2 Register (MCR2)

The multichannel control 2 register (MCR2) is shown in Figure 15-76 and described in Table 15-85.

Figure 15-76. Multichannel Control 2 Register (MCR2)

15	10	9	8
Reserved		XMCME	XPBBLK
R-0		R/W-0	R/W-0

7	6	5	4	2	1	0
XPBBLK	XPABLK		XCBLK		XMCM	
R/W-0	R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-85. Multichannel Control 2 Register (MCR2) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
9	XMCME		Transmit multichannel partition mode bit. XMCME determines whether only 32 channels or all 128 channels are to be individually selectable. XMCME is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero).
		0	2-partition mode. Only partitions A and B are used. You can control up to 32 channels in the transmit multichannel selection mode selected with the XMCM bits. If XMCM = 01b or 10b, assign 16 channels to partition A with the XPABLK bits. Assign 16 channels to partition B with the XPBBLK bits. If XMCM = 11b(for symmetric transmission and reception), assign 16 channels to receive partition A with the RPABLK bits. Assign 16 channels to receive partition B with the RPBBLK bits. You control the channels with the appropriate transmit channel enable registers: XCERA: Channels in partition A XCERB: Channels in partition B
		1	8-partition mode. All partitions (A through H) are used. You can control up to 128 channels in the transmit multichannel selection mode selected with the XMCM bits. You control the channels with the appropriate transmit channel enable registers: XCERA: Channels 0 through 15 XCERB: Channels 16 through 31 XCERC: Channels 32 through 47 XCERD: Channels 48 through 63 XCERE: Channels 64 through 79 XCERF: Channels 80 through 95 XCERG: Channels 96 through 111 XCERH: Channels 112 through 127

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-85. Multichannel Control 2 Register (MCR2) Field Descriptions (continued)

Bit	Field	Value	Description
8-7	XPBBLK	0-3h	Transmit partition B block bits XPBBLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to partitions A and B of the transmitter. The 128 transmit channels of the McBSP are divided equally among 8 blocks (0 through 7). When XPBBLK is applicable, use XPBBLK to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B, as shown in the following table. Use the XPABLK bit to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A. If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the transmitter is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The XCBLK bits are regularly updated to indicate which block is active. When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).
		0	Block 1: channels 16 through 31
		1h	Block 3: channels 48 through 63
		2h	Block 5: channels 80 through 95
		3h	Block 7: channels 112 through 127
6-5	XPABLK		Transmit partition A block bits. XPABLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to partitions A and B of the transmitter. See the description for XPBBLK (bits 8-7) for more information about assigning blocks to partitions A and B.
		0	Block 0: channels 0 through 15
		1h	Block 2: channels 32 through 47
		2h	Block 4: channels 64 through 79
		3h	Block 6: channels 96 through 111
4-2	XCBLK		Transmit current block indicator. XCBLK indicates which block of 16 channels is involved in the current McBSP transmission:
		0	Block 0: channels 0 through 15
		1h	Block 1: channels 16 through 31
		2h	Block 2: channels 32 through 47
		3h	Block 3: channels 48 through 63
		4h	Block 4: channels 64 through 79
		5h	Block 5: channels 80 through 95
		6h	Block 6: channels 96 through 111
		7h	Block 7: channels 112 through 127
1-0	XMCM	0-3h	Transmit multichannel selection mode bits. XMCM determines whether all channels or only selected channels are enabled and unmasked for transmission. For more details on how the channels are affected, see Section 15.6.7 *Transmit Multichannel Selection Modes*.
		0	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.
		1h	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked. The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.
		2h	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERs.
		3h	This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.

Copyright © 2012–2019, Texas Instruments Incorporated

15.12.9 Pin Control Register (PCR)

Each McBSP has one pin control register (PCR). Table 15-86 describes the bits of PCR. This register enables you to:

- Choose a frame-synchronization mode for the transmitter (FSXM) and for the receiver (FSRM)
- Choose a clock mode for transmitter (CLKXM) and for the receiver (CLKRM)
- Select the input clock source for the sample rate generator (SCLKME, in conjunction with the CLKSM bit of SRGR2)
- Choose whether frame-synchronization signals are active low or active high (FSXP for transmission, FSRP for reception)
- Specify whether data is sampled on the falling edge or the rising edge of the clock signals (CLKXP for transmission, CLKRP for reception)

The pin control register (PCR) is shown in Figure 15-77 and described in Table 15-86.

Figure 15-77. Pin Control Register (PCR)

15	12	11	10	9	8
Reserved		FSXM	FSRM	CLKXM	CLKRM
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	4	3	2	1	0
SCLKME	Reserved		FSXP	FSRP	CLKXP	CLKRP
R/W-0	R-0		R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-86. Pin Control Register (PCR) Field Descriptions

Bit	Field	Value	Description
15:12	Reserved	0	Reserved bit (not available for your use). It is a read-only bit and returns a 0 when read.
11	FSXM		Transmit frame-synchronization mode bit. FSXM determines whether transmit frame-synchronization pulses are supplied externally or internally. The polarity of the signal on the FSX pin is determined by the FSXP bit.
		0	Transmit frame synchronization is supplied by an external source via the FSX pin.
		1	Transmit frame synchronization is generated internally by the Sample Rate generator, as determined by the FSGM bit of SRGR2.
10	FSRM		Receive frame-synchronization mode bit. FSRM determines whether receive frame-synchronization pulses are supplied externally or internally. The polarity of the signal on the FSR pin is determined by the FSRP bit.
		0	Receive frame synchronization is supplied by an external source via the FSR pin.
		1	Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-86. Pin Control Register (PCR) Field Descriptions (continued)

Bit	Field	Value	Description
9	CLKXM		Transmit clock mode bit. CLKXM determines whether the source for the transmit clock is external or internal, and whether the MCLKX pin is an input or an output. The polarity of the signal on the MCLKX pin is determined by the CLKXP bit. In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master, make sure that CLKX is an output. If the McBSP is a slave, make sure that CLKX is an input. Not in clock stop mode (CLKSTP = 00b or 01b):
		0	The transmitter gets its clock signal from an external source via the MCLKX pin.
		1	Internal CLKX is driven by the sample rate generator of the McBSP. The MCLKX pin is an output pin that reflects internal CLKX.
			In clock stop mode (CLKSTP = 10b or 11b):
		0	The McBSP is a slave in the SPI protocol. The internal transmit clock (CLKX) is driven by the SPI master via the MCLKX pin. The internal receive clock (MCLKR) is driven internally by CLKX, so that both the transmitter and the receiver are controlled by the external master clock.
		1	The McBSP is a master in the SPI protocol. The sample rate generator drives the internal transmit clock (CLKX). Internal CLKX is reflected on the MCLKX pin to drive the shift clock of the SPI-compliant slaves in the system. Internal CLKX also drives the internal receive clock (MCLKR), so that both the transmitter and the receiver are controlled by the internal master clock.
8	CLKRM		Receive clock mode bit. The role of CLKRM and the resulting effect on the MCLKR pin depend on whether the McBSP is in the digital loopback mode (DLB = 1). The polarity of the signal on the MCLKR pin is determined by the CLKRP bit. Not in digital loopback mode (DLB = 0):
		0	The MCLKR pin is an input pin that supplies the internal receive clock (MCLKR).
		1	Internal MCLKR is driven by the sample rate generator of the McBSP. The MCLKR pin is an output pin that reflects internal MCLKR.
			In digital loopback mode (DLB = 1):
		0	The MCLKR pin is in the high impedance state. The internal receive clock (MCLKR) is driven by the internal transmit clock (CLKX). CLKX is derived according to the CLKXM bit.
		1	Internal MCLKR is driven by internal CLKX. The MCLKR pin is an output pin that reflects internal MCLKR. CLKX is derived according to the CLKXM bit.
7	SCLKME		Sample rate generator input clock mode bit. The sample rate generator can produce a clock signal, CLKG. The frequency of CLKG is: CLKG freq. = (Input clock frequency) / (CLKGDV + 1) SCLKME is used in conjunction with the CLKSM bit to select the input clock. **SCLKME** / **CLKSM** / **Input Clock For Sample Rate Generator** 0 / 0 / Reserved 0 / 1 / LSPCLK The input clock for the sample rate generator is taken from the MCLKR pin or from the MCLKX pin, depending on the value of the CLKSM bit of SRGR2: **SCLKME** / **CLKSM** / **Input Clock For Sample Rate Generator** 1 / 0 / Signal on MCLKR pin 1 / 1 / Signal on MCLKX pin
6-4	Reserved		Reserved
3	FSXP		Transmit frame-synchronization polarity bit. FSXP determines the polarity of FSX as seen on the FSX pin.
		0	Transmit frame-synchronization pulses are active high.
		1	Transmit frame-synchronization pulses are active low.
2	FSRP		Receive frame-synchronization polarity bit. FSRP determines the polarity of FSR as seen on the FSR pin.
		0	Receive frame-synchronization pulses are active high.
		1	Receive frame-synchronization pulses are active low.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-86. Pin Control Register (PCR) Field Descriptions (continued)

Bit	Field	Value	Description
1	CLKXP		Transmit clock polarity bit. CLKXP determines the polarity of CLKX as seen on the MCLKX pin.
		0	Transmit data is sampled on the rising edge of CLKX.
		1	Transmit data is sampled on the falling edge of CLKX.
0	CLKRP		Receive clock polarity bit. CLKRP determines the polarity of CLKR as seen on the MCLKR pin.
		0	Receive data is sampled on the falling edge of MCLKR.
		1	Receive data is sampled on the rising edge of MCLKR.

Table 15-87. Pin Configuration

Pin	Selected as Output When …	Selected as Input When …
CLKX	CLKXM = 1	CLKXM = 0
FSX	FSXM = 1	FSXM = 0
CLKR	CLKRM = 1	CLKRM = 0
FSR	FSRM = 1	FSRM = 0

15.12.10 Receive Channel Enable Registers (RCERA, RCERB, RCERC, RCERD, RCERE, RCERF, RCERG, RCERH)

Each McBSP has eight receive channel enable registers of the format shown in Figure 15-78. There is one enable register for each of the receive partitions: A, B, C, D, E, F, G, and H. Table 15-88 provides a summary description that applies to any bit x of a receive channel enable register.

These memory-mapped registers are only used when the receiver is configured to allow individual enabling and disabling of the channels (RMCM = 1). For more details about the way these registers are used, see Section 15.12.10.1, *RCERs Used in the Receive Multichannel Selection Mode*.

The receive channel enable registers (RCERA...RCERH) are shown in Figure 15-78 and described in Table 15-88.

Figure 15-78. Receive Channel Enable Registers (RCERA...RCERH)

15	14	13	12	11	10	9	8
RCE15	RCE14	RCE13	RCE12	RCE11	RCE10	RCE9	RCE8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
RCE7	RCE6	RCE5	RCE4	RCE3	RCE2	RCE1	RCE0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-88. Receive Channel Enable Registers (RCERA...RCERH) Field Descriptions

Bit	Field	Value	Description
15-0	RCEx		Receive channel enable bit.
			For receive multichannel selection mode (RMCM = 1):
		0	Disable the channel that is mapped to RCEx.
		1	Enable the channel that is mapped to RCEx.

15.12.10.1 RCERs Used in the Receive Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the RCERs depends on whether 32 or 128 channels are individually selectable, as defined by the RMCME bit. For each of these two cases, Table 15-89 shows which block of channels is assigned to each of the RCERs used. For each RCER, the table shows which channel is assigned to each of the bits.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-89. Use of the Receive Channel Enable Registers

Number of Selectable Channels	Block Assignments		Channel Assignments	
	RCERx	Block Assigned	Bit in RCERx	Channel Assigned
32 (RMCME = 0)	RCERA	Channels n to (n + 15)	RCE0	Channel n
			RCE1	Channel (n + 1)
			RCE2	Channel (n + 2)
			:	:
		The block of channels is chosen with the RPABLK bits.	RCE15	Channel (n + 15)
	RCERB	Channels m to (m + 15)	RCE0	Channel m
			RCE1	Channel (m + 1)
			RCE2	Channel (m + 2)
			:	:
		The block of channels is chosen with the RPBBLK bits.	RCE15	Channel (m + 15)
128 (RMCME = 1)	RCERA	Block 0	RCE0	Channel 0
			RCE1	Channel 1
			RCE2	Channel 2
			:	:
			RCE15	Channel 15
	RCERB	Block 1	RCE0	Channel 16
			RCE1	Channel 17
			RCE2	Channel 18
			:	:
			RCE15	Channel 31
	RCERC	Block 2	RCE0	Channel 32
			RCE1	Channel 33
			RCE2	Channel 34
			:	:
			RCE15	Channel 47
	RCERD	Block 3	RCE0	Channel 48
			RCE1	Channel 49
			RCE2	Channel 50
			:	:
			RCE15	Channel 63
	RCERE	Block 4	RCE0	Channel 64
			RCE1	Channel 65
			RCE2	Channel 66
			:	:
			RCE15	Channel 79
	RCERF	Block 5	RCE0	Channel 80
			RCE1	Channel 81
			RCE2	Channel 82
			:	:
			RCE15	Channel 95
	RCERG	Block 6	RCE0	Channel 96
			RCE1	Channel 97
			RCE2	Channel 98
			:	:
			RCE15	Channel 111

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-89. Use of the Receive Channel Enable Registers (continued)

Number of Selectable Channels	Block Assignments		Channel Assignments	
	RCERx	Block Assigned	Bit in RCERx	Channel Assigned
	RCERH	Block 7	RCE0	Channel 112
			RCE1	Channel 113
			RCE2	Channel 114
			:	:
			RCE15	Channel 127

15.12.11 Transmit Channel Enable Registers (XCERA, XCERB, XCERC, XCERD, XCERE, XCERF, XCERG, XCERH)

Each McBSP has eight transmit channel enable registers of the form shown in Figure 15-79. There is one for each of the transmit partitions: A, B, C, D, E, F, G, and H. Table 15-90 provides a summary description that applies to each bit XCEx of a transmit channel enable register.

The XCERs are only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (XMCM is nonzero).

The transmit channel enable registers (XCERA...XCERH) are shown in Figure 15-79 and described in Table 15-90.

Figure 15-79. Transmit Channel Enable Registers (XCERA...XCERH)

15	14	13	12	11	10	9	8
XCE15	XCE14	XCE13	XCE12	XCE11	XCE10	XCE9	XCE8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
XCE7	XCE6	XCE5	XCE4	XCE3	XCE2	XCE1	XCE0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 15-90. Transmit Channel Enable Registers (XCERA...XCERH) Field Descriptions

Bit	Field	Value	Description
15-0	XCEx		Transmit channel enable bit. The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.
			For multichannel selection when XMCM = 01b (all channels disabled unless selected):
		0	Disable and mask the channel that is mapped to XCEx.
		1	Enable and unmask the channel that is mapped to XCEx.
			For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):
		0	Mask the channel that is mapped to XCEx.
		1	Unmask the channel that is mapped to XCEx.
			For multichannel selection when XMCM = 11b (all channels masked unless selected):
		0	Mask the channel that is mapped to XCEx. Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.
		1	Unmask the channel that is mapped to XCEx. If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur.

Copyright © 2012–2019, Texas Instruments Incorporated

15.12.11.1 XCERs Used in a Transmit Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the XCERs depends on whether 32 or 128 channels are individually selectable, as defined by the XMCME bit. These two cases are shown in Table 15-91. The table shows which block of channels is assigned to each XCER that is used. For each XCER, the table shows which channel is assigned to each of the bits.

NOTE: When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive channel enable registers (RCERs) to enable channels and uses the XCERs to unmask channels for transmission.

Table 15-91. Use of the Transmit Channel Enable Registers

Number of Selectable Channels	Block Assignments		Channel Assignments	
	XCERx	Block Assigned	Bit in XCERx	Channel Assigned
32 (XMCME = 0)	XCERA	Channels n to (n + 15)	XCE0	Channel n
			XCE1	Channel (n + 1)
			XCE2	Channel (n + 2)
			:	:
		When XMCM = 01b or 10b, the block of channels is chosen with the XPABLK bits. When XMCM = 11b, the block is chosen with the RPABLK bits.	XCE15	Channel (n + 15)
	XCERB	Channels m to (m + 15)	XCE0	Channel m
			XCE1	Channel (m + 1)
			XCE2	Channel (m + 2)
			:	:
		When XMCM = 01b or 10b, the block of channels is chosen with the XPBBLK bits. When XMCM = 11b, the block is chosen with the RPBBLK bits.	XCE15	Channel (m + 15)
128 (XMCME = 1)	XCERA	Block 0	XCE0	Channel 0
			XCE1	Channel 1
			XCE2	Channel 2
			:	:
			XCE15	Channel 15
	XCERB	Block 1	XCE0	Channel 16
			XCE1	Channel 17
			XCE2	Channel 18
			:	:
			XCE15	Channel 31
	XCERC	Block 2	XCE0	Channel 32
			XCE1	Channel 33
			XCE2	Channel 34
			:	:
			XCE15	Channel 47
	XCERD	Block 3	XCE0	Channel 48
			XCE1	Channel 49
			XCE2	Channel 50
			:	:
			XCE15	Channel 63

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-91. Use of the Transmit Channel Enable Registers (continued)

Number of Selectable Channels	Block Assignments		Channel Assignments	
	XCERx	Block Assigned	Bit in XCERx	Channel Assigned
	XCERE	Block 4	XCE0	Channel 64
			XCE1	Channel 65
			XCE2	Channel 66
			:	:
			XCE15	Channel 79
	XCERF	Block 5	XCE0	Channel 80
			XCE1	Channel 81
			XCE2	Channel 82
			:	:
			XCE15	Channel 95
	XCERG	Block 6	XCE0	Channel 96
			XCE1	Channel 97
			XCE2	Channel 98
			:	:
			XCE15	Channel 111
	XCERH	Block 7	XCE0	Channel 112
			XCE1	Channel 113
			XCE2	Channel 114
			:	:
			XCE15	Channel 127

15.12.12 Interrupt Generation

McBSP registers can be programmed to receive and transmit data through DRR2/DRR1 and DXR2/DXR1 registers, respectively. The CPU can directly access these registers to move data from memory to these registers. Interrupt signals will be based on these register pair contents and its related flags.MRINT/MXINT will generate CPU interrupts for receive and transmit conditions.

15.12.12.1 McBSP Receive Interrupt Generation

In the McBSP module, data receive and error conditions generate two sets of interrupt signals. One set is used for the CPU and the other set is for DMA.

Figure 15-80. Receive Interrupt Generation

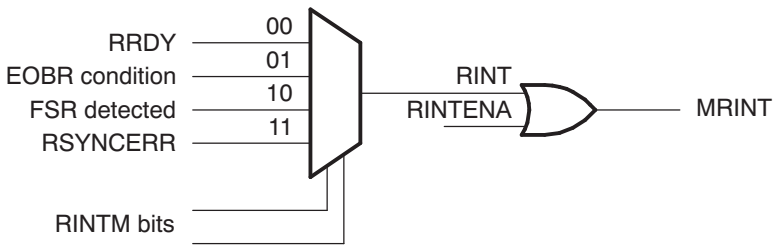


Table 15-92. Receive Interrupt Sources and Signals

McBSP Interrupt Signal	Interrupt Flags	Interrupt Enables in SPCR1	Interrupt Enables	Type of Interrupt	Interrupt Line
		RINTM Bits			

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-92. Receive Interrupt Sources and Signals (continued)

McBSP Interrupt Signal	Interrupt Flags	Interrupt Enables in SPCR1	Interrupt Enables	Type of Interrupt	Interrupt Line
RINT	RRDY	0	RINTENA	Every word receive	MRINT
	EOBR	1	RINTENA	Every 16 channel block boundary	
	FSR	10	RINTENA	On every FSR	
	RSYNCERR	11	RINTENA	Frame sync error	

NOTE: Since X/RINT, X/REVTA, and X/RXFFINT share the same CPU interrupt, it is recommended that all applications use one of the above selections for interrupt generation. If multiple interrupt enables are selected at the same time, there is a likelihood of interrupts being masked or not recognized.

15.12.12.2 McBSP Transmit Interrupt Generation

McBSP module data transmit and error conditions generate two sets of interrupt signals. One set is used for the CPU and the other set is for DMA.

Figure 15-81. Transmit Interrupt Generation

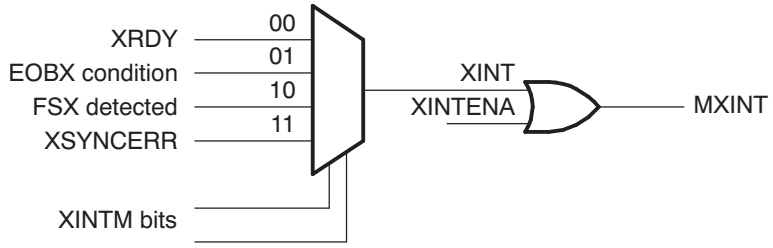


Table 15-93. Transmit Interrupt Sources and Signals

McBSP Interrupt Signal	Interrupt Flags	Interrupt Enables in SPCR2	Interrupt Enables	Type of Interrupt	Interrupt Line
		XINTM Bits			
XINT	XRDY	0	XINTENA	Every word transmit	MXINT
	EOBX	1	XINTENA	Every 16-channel block boundary	
	FSX	10	XINTENA	On every FSX	
	XSYNCERR	11	XINTENA	Frame sync error	

15.12.12.3 Error Flags

The McBSP has several error flags both on receive and transmit channel. Table 15-94 explains the error flags and their meaning.

Table 15-94. Error Flags

Error Flags	Function
RFULL	Indicates DRR2/DRR1 are not read and RXR register is overwritten
RSYNCERR	Indicates unexpected frame-sync condition, current data reception will abort and restart. Use RINTM bit 11 for interrupt generation on this condition.
XSYNCERR	Indicates unexpected frame-sync condition, current data transmission will abort and restart. Use XINTM bit 11 for interrupt generation on this condition.

Copyright © 2012–2019, Texas Instruments Incorporated

15.12.12.4 McBSP Interrupt Enable Register

Figure 15-82. McBSP Interrupt Enable Register (MFFINT)

15				8
Reserved				
R-0				
7	**3**	**2**	**1**	**0**
Reserved		RINT ENA	Reserved	XINT ENA
R-0		R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-95. McBSP Interrupt Enable Register (MFFINT) Field Descriptions

Bit	Field	Value	Description
15:3	Reserved		Reserved
2	RINT ENA		Enable for Receive Interrupt
		0	Receive interrupt on RRDY is disabled.
		1	Receive interrupt on RRDY is enabled.
1	Reserved		
0	XINT ENA		Enable for transmit Interrupt
		0	Transmit interrupt on XRDY is disabled.
		1	Transmit interrupt on XRDY is enabled.

15.12.12.5 McBSP Modes

McBSP, in its normal mode, communicates with various types of Codecs with variable word size. Apart from this mode, the McBSP uses time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices. The multichannel mode provides flexibility while transmitting/receiving selected channels or all the channels in a TDM stream.

Table 15-96 provides a quick reference to McBSP mode selection.

Table 15-96. McBSP Mode Selection

		Register Bits Used for Mode Selection				
		MCR1 bit 9,0		MCR2 bit 9,1,0		
No.	McBSP Word Size	RMCME	RMCM	XMCME	XMCM	Mode and Function Description
						Normal Mode
1	8/12/16/20/24/32 bit words	0	0	0	0	All types of Codec interface will use this selection
						Multichannel Mode
2	8-bit words					2 Partition or 32-channel Mode
		0	1	0	1	All channels are disabled,unless selected in X/RCERA/B
		0	1	0	10	All channels are enabled,but masked unless selected in X/RCERA/B
		0	1	0	11	Symmetric transmit, receive 8 Partition or 128 Channel Mode Transmit/ Receive Channels selected by X/RCERA to X/RCERH bits
						Multichannel Mode is ON
		1	1	1	1	All channels are disabled,unless selected in

Copyright © 2012–2019, Texas Instruments Incorporated

Table 15-96. McBSP Mode Selection (continued)

No.	McBSP Word Size	Register Bits Used for Mode Selection				Mode and Function Description
		MCR1 bit 9,0		MCR2 bit 9,1,0		
		RMCME	RMCM	XMCME	XMCM	
						XCERs
		1	1	1	10	All channels are enabled,but masked unless selected in XCERs
		1	1	1	11	Symmetric transmit, receive
						Continuous Mode - Transmit
		1	0	1	0	**Multi-Channel Mode is OFF**
						All 128 channels are active and enabled

Copyright © 2012–2019, Texas Instruments Incorporated

M3 Micro Direct Memory Access (µDMA)

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

Topic **Page**

16.1 Overview **1149**
16.2 Block Diagram **1149**
16.3 Functional Description **1150**
16.4 Initialization and Configuration **1164**
16.5 Register Map **1169**
16.6 µDMA Channel Control Structure **1171**
16.7 µDMA Register Descriptions **1175**

Copyright © 2012–2019, Texas Instruments Incorporated

16.1 Overview

The µDMA controller provides the following features:

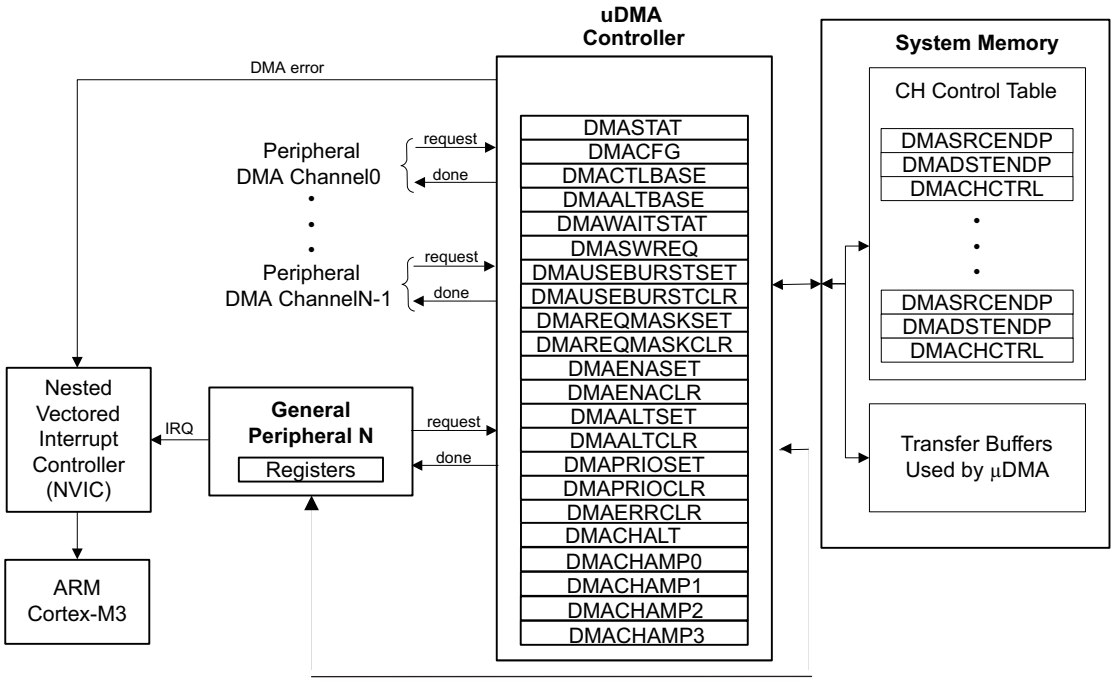
- ARM PrimeCell® 32-channel configurable µDMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of arbitrary transfers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

16.2 Block Diagram

Figure 16-1 illustrates the µDMA block diagram.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-1. µDMA Block Diagram



16.3 Functional Description

The µDMA controller is a flexible and highly configurable DMA controller designed to work efficiently with the microcontroller's Cortex-M3 processor core. It supports multiple data sizes and address increment schemes, multiple levels of priority among DMA channels, and several transfer modes to allow for sophisticated programmed data transfers. The µDMA controller's usage of the bus is always subordinate to the processor core, so it never holds up a bus transaction by the processor. Because the µDMA controller is only using otherwise-idle bus cycles, the data transfer bandwidth it provides is essentially free, with no impact on the rest of the system. The bus architecture has been optimized to greatly enhance the ability of the processor core and the µDMA controller to efficiently share the on-chip bus, thus improving performance. The optimizations include RAM striping and peripheral bus segmentation, which in many cases allow both the processor core and the µDMA controller to access the bus and perform simultaneous data transfers.

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

Each peripheral function that is supported has a dedicated channel on the µDMA controller that can be configured independently. The µDMA controller implements a unique configuration method using channel control structures that are maintained in system memory by the processor. While simple transfer modes are supported, it is also possible to build up sophisticated "task" lists in memory that allow the µDMA controller to perform arbitrary-sized transfers to and from arbitrary locations as part of a single transfer request. The µDMA controller also supports the use of ping-pong buffering to accommodate constant streaming of data to or from a peripheral.

Each channel also has a configurable arbitration size. The arbitration size is the number of items that are transferred in a burst before the µDMA controller rearbitrates for channel priority. Using the arbitration size, it is possible to control exactly how many items are transferred to or from a peripheral each time it makes a µDMA service request.

Copyright © 2012–2019, Texas Instruments Incorporated

16.3.1 Channel Assignments

µDMA channels 0-31 are assigned to different peripherals according to the following table. The DMA Channel Map (DMACHMAPx) registers can be used to specify the first, second, or third channel mapping assignment.primary or secondary assignment.

NOTE: Channels noted in the table as "Available for software" may be assigned to peripherals in the future. However, they are currently available for software use. Channel 30 is dedicated for software use.

The USB endpoints mapped to µDMA channels 0-3 can be changed with the USBDMASEL register.

Because of the way the µDMA controller interacts with peripherals, the µDMA channel for the peripheral must be enabled in order for the µDMA controller to be able to read and write the peripheral registers, even if a different µDMA channel is used to perform the µDMA transfer. To minimize confusion and chance of software errors, it is best practice to use a peripheral's µDMA channel for performing all µDMA transfers for that peripheral, even if it is processor-triggered and using AUTO mode, which could be considered a software transfer. Note that if the software channel is used, interrupts occur on the dedicated µDMA interrupt vector. If the peripheral channel is used, then the interrupt occurs on the interrupt vector for the peripheral.

Table 16-1. µDMA Channel Assignment Mapping

DMACHALT Encoding	0	1	
DMACHMAPx Encoding	0	1	2
µDMA Channel	First Assignment	Second Assignment	Third Assignment
0	USB Endpoint 1 Receive	UART2 Receive	Reserved
1	USB Endpoint 1 Transmit	UART2 Transmit	Reserved
2	USB Endpoint 2 Receive	General-Purpose Timer 3A	Reserved
3	USB Endpoint 2 Transmit	General-Purpose Timer 3B	Reserved
4	USB Endpoint 3 Receive	General-Purpose Timer 2A	Reserved
5	USB Endpoint 3 Transmit	General-Purpose Timer 2B	Reserved
6	Ethernet Receive	General-Purpose Timer 2A	Reserved
7	Ethernet Transmit	General-Purpose Timer 2B	Reserved
8	UART0 Receive	UART1 Receive	Reserved
9	UART0 Transmit	UART1 Transmit	Reserved
10	SSI0 Receive	SSI1 Receive	Reserved
11	SSI0 Transmit	SSI1 Transmit	Reserved
12	Available for software	UART2 Receive	SSI2 Receive
13	Available for software	UART2 Transmit	SSI2 Transmit
14	Reserved	General-Purpose Timer 2A	SSI3 Receive
15	Reserved	General-Purpose Timer 2B	SSI3 Transmit
16	Reserved	Available for software	UART3 Receive
17	Reserved	Available for software	UART3 Transmit
18	General-Purpose Timer 0A	General-Purpose Timer 1A	UART4 Receive
19	General-Purpose Timer 0B	General-Purpose Timer 1B	UART4 Transmit
20	General-Purpose Timer 1A	Adv GPIO / EPI0RX (EPI Non-blocking read FIFO)	Reserved
21	General-Purpose Timer 1B	Adv GPIO / EPI0TX (EPI Write FIFO)	Reserved
22	UART1 Receive	Available for software	Reserved
23	UART1 Transmit	Available for software	Reserved
24	SSI1 Receive	Reserved	Reserved
25	SSI1 Transmit	Reserved	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-1. µDMA Channel Assignment Mapping (continued)

DMACHALT Encoding	0	1	
DMACHMAPx Encoding	0	1	2
µDMA Channel	First Assignment	Second Assignment	Third Assignment
26	Available for software	Reserved	Reserved
27	Available for software	Reserved	Reserved
28	Reserved	Available for software	Reserved
29	Reserved	Available for software	Reserved
30	Dedicated for software use		Reserved
31	Reserved		Reserved

NOTE: There is another register, DMACHALT, which controls the channel assignments for the First and Second mapping Only (not Third Mapping). This is for compatibility with Stellaris devices. The register (DMACHALT vs DMACHMAPx) which is written later, controls the channel assignment mapping.

16.3.2 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.

16.3.3 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.

16.3.4 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. Table 16-2 shows how each peripheral supports the two request types.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-2. Request Type Support

Peripheral	Single Request Signal	Burst Request Signal
EPI WFIFO	None	WFIFO Level (configurable)
EPI NBRFIFO	None	NBRFIFO Level (configurable)
Ethernet TX	TX FIFO empty	None
Ethernet RX	RX packet received	None
General-Purpose Timer	Raw interrupt pulse	None
SSI TX	TX FIFO Not Full	TX FIFO Level (fixed at 4)
SSI RX	RX FIFO Not Empty	RX FIFO Level (fixed at 4)
UART TX	TX FIFO Not Full	TX FIFO Level (configurable)
UART RX	RX FIFO Not Empty	RX FIFO Level (configurable)
USB TX	None	FIFO TXRDY
USB RX	None	FIFO RXRDY

16.3.4.1 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.

16.3.4.2 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.

16.3.5 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 16-3 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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-3. 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 16-4 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 16-4. 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 Section 16.6. 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.

16.3.6 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.

16.3.6.1 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.

Copyright © 2012–2019, Texas Instruments Incorporated

16.3.6.2 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.

16.3.6.3 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.

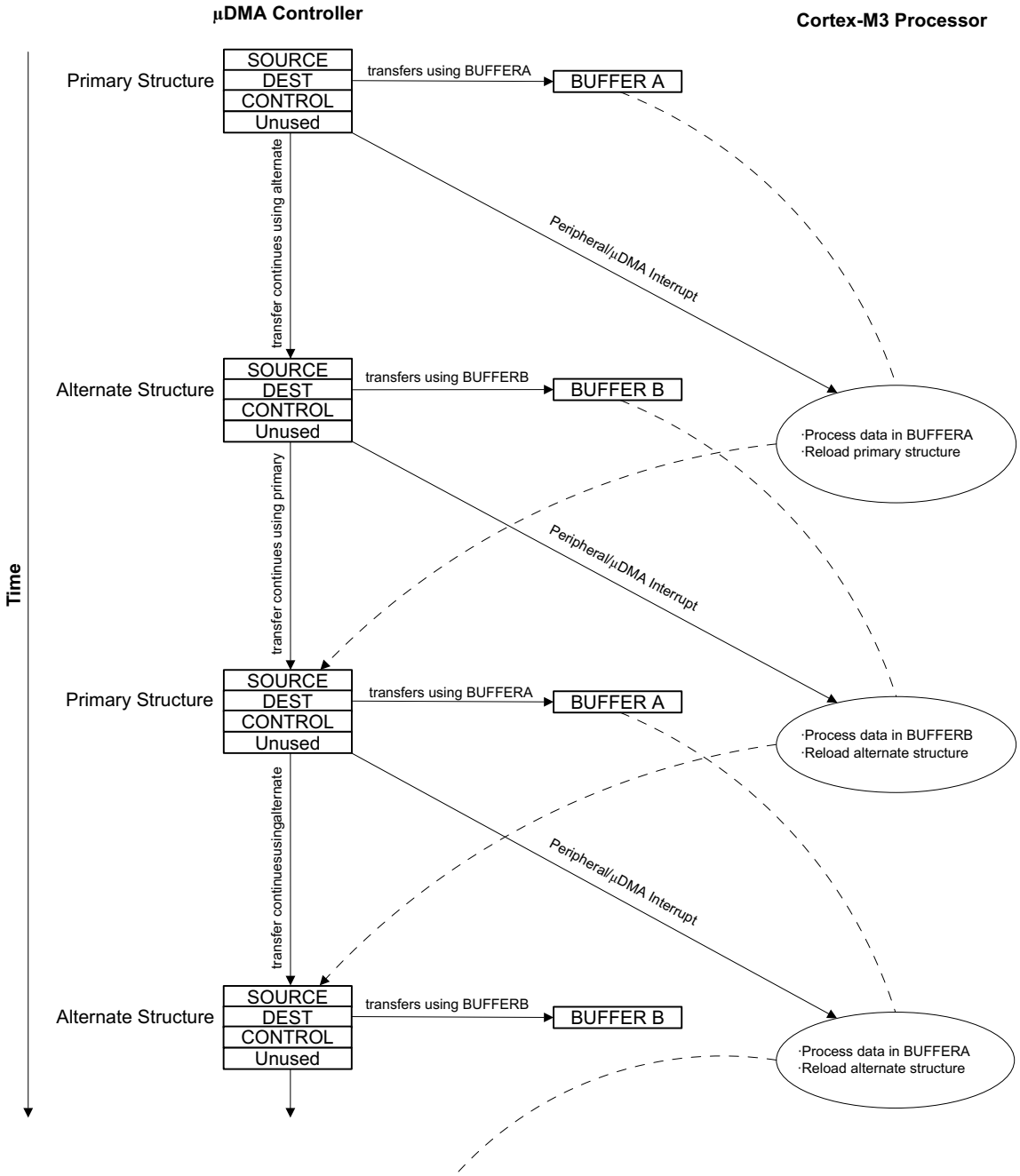
16.3.6.4 Ping-Pong

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.

Figure 16-2 provides an example operation in Ping-Pong mode.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-2. Example of Ping-Pong µDMA Transaction



16.3.6.5 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

Copyright © 2012–2019, Texas Instruments Incorporated

between using the primary control structure to copy the next transfer instruction from the list and then executing the new transfer instruction. The end of the list is marked by programming the control word for the last entry to use Basic transfer mode. Once the last transfer is performed using Basic mode, the µDMA controller stops. A completion interrupt is generated only after the last transfer. It is possible to loop the list by having the last entry copy the primary control structure to point back to the beginning of the list (or to a new list). It is also possible to trigger a set of other channels to perform a transfer, either directly, by programming a write to the software trigger for another channel, or indirectly, by causing a peripheral action that results in a µDMA request.

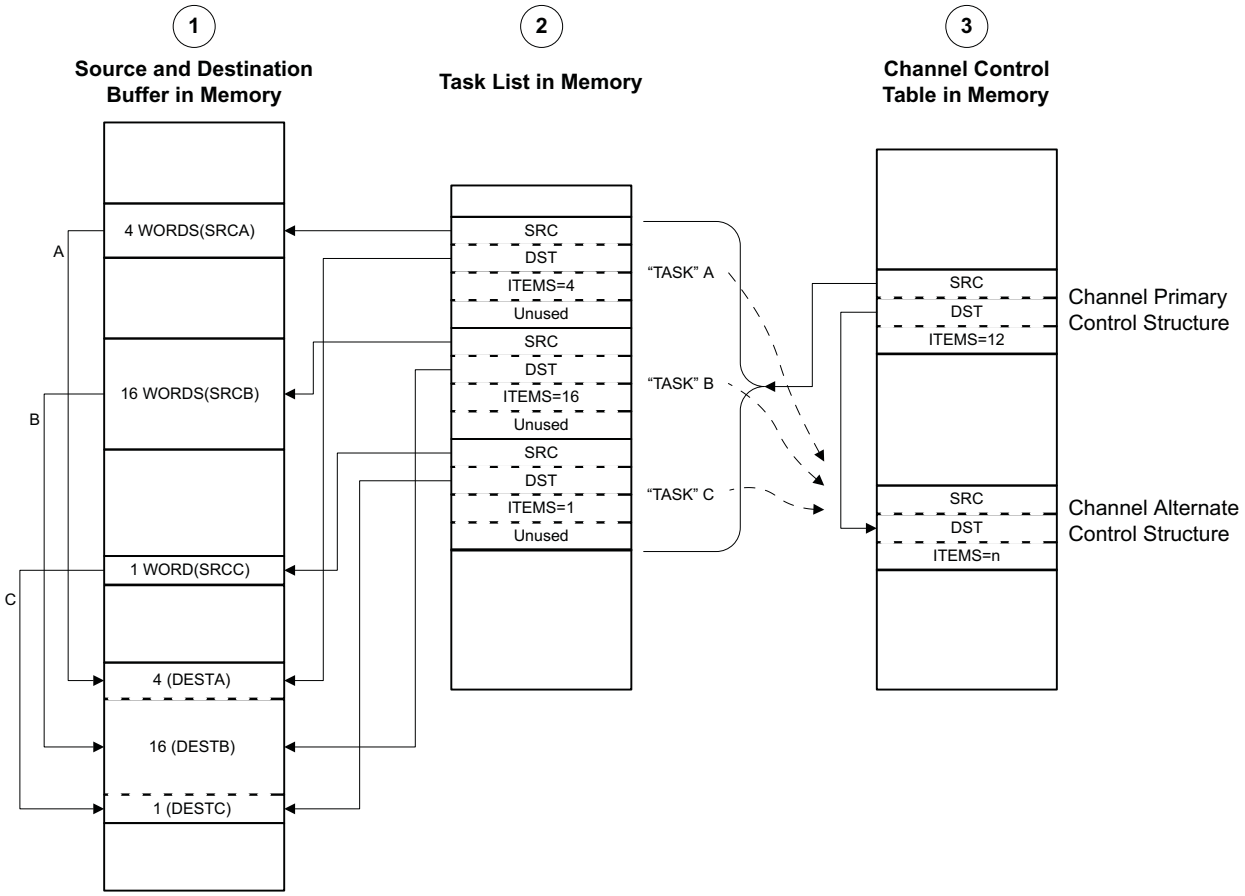
By programming the µDMA controller using this method, a set of arbitrary transfers can be performed based on a single µDMA request.

Figure 16-3 and Figure 16-4 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 16-3 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 16-4 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.

Copyright © 2012–2019, Texas Instruments Incorporated

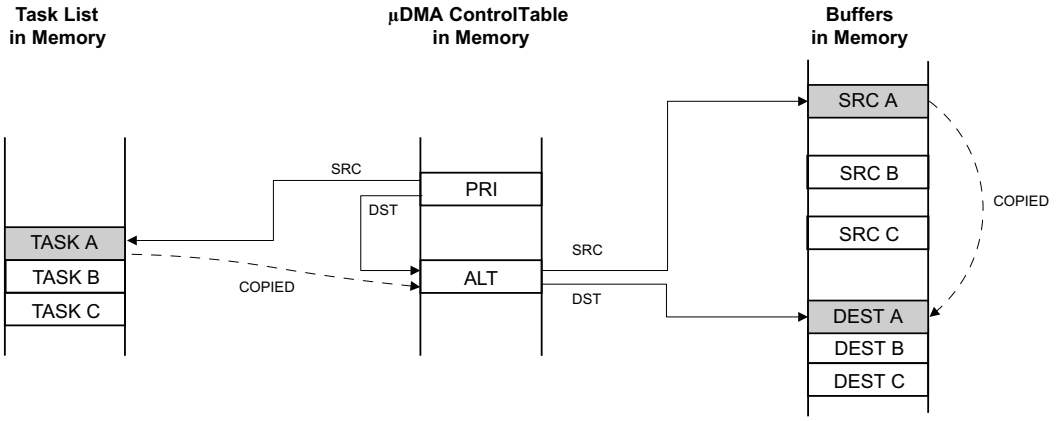
Figure 16-3. Memory Scatter-Gather, Setup and Configuration



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 "tasklist" 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.

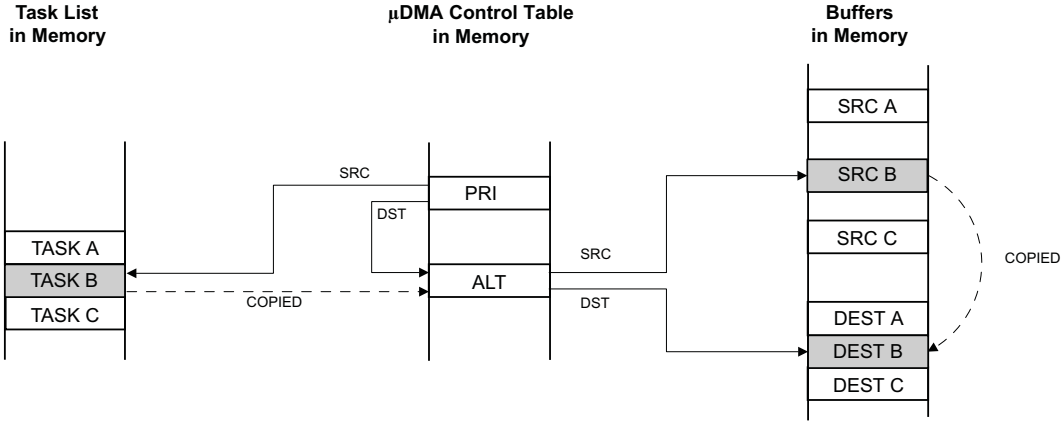
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-4. Memory Scatter-Gather, µDMA Copy Sequence



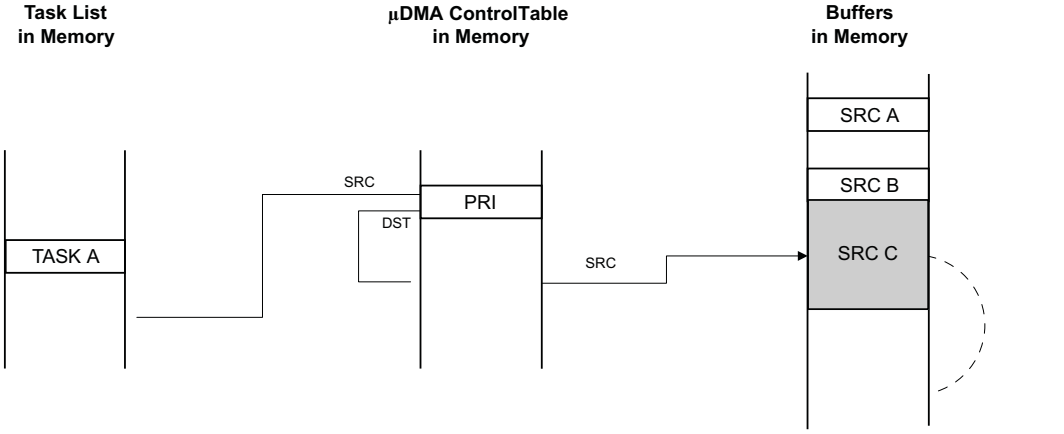
Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer A to the destination buffer.



Using thechannel'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 thechannel'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.

Copyright © 2012–2019, Texas Instruments Incorporated

16.3.6.6 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.

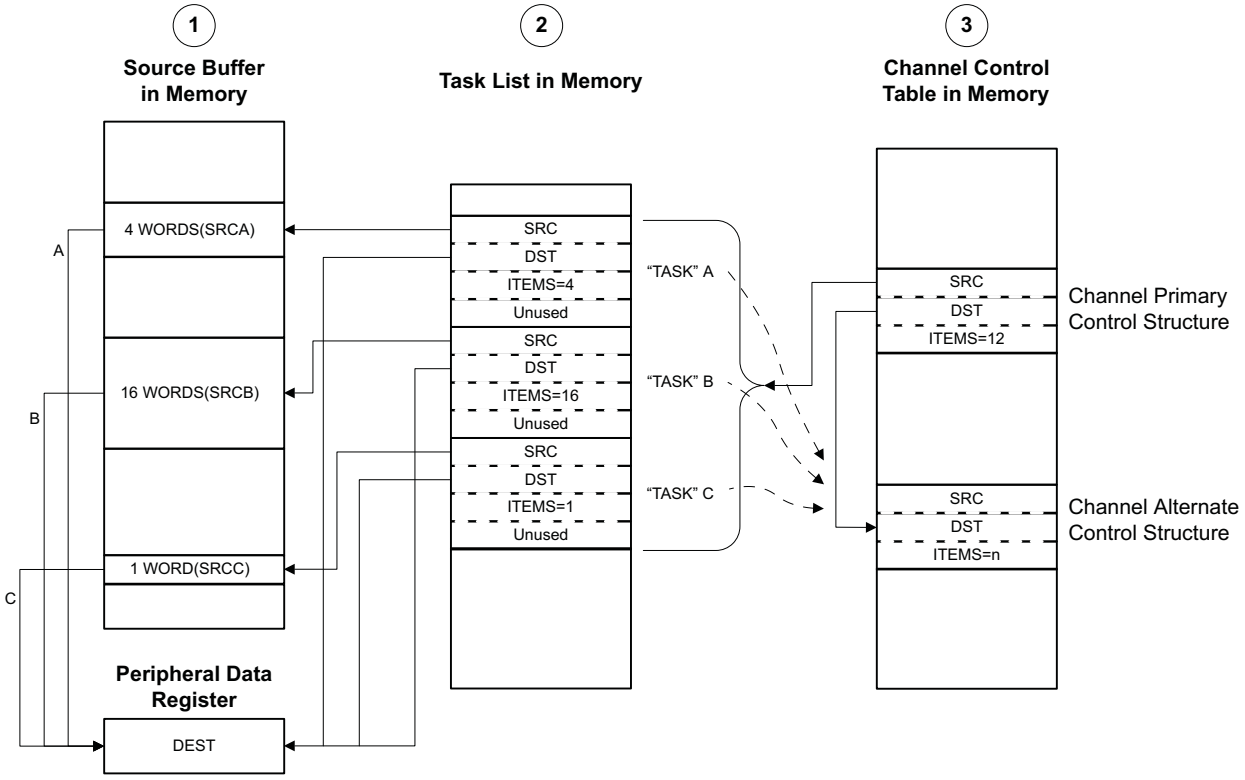
By using this method, the µDMA controller can transfer data to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

Figure 16-5 and Figure 16-6 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 16-5 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 16-6 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.

Copyright © 2012–2019, Texas Instruments Incorporated

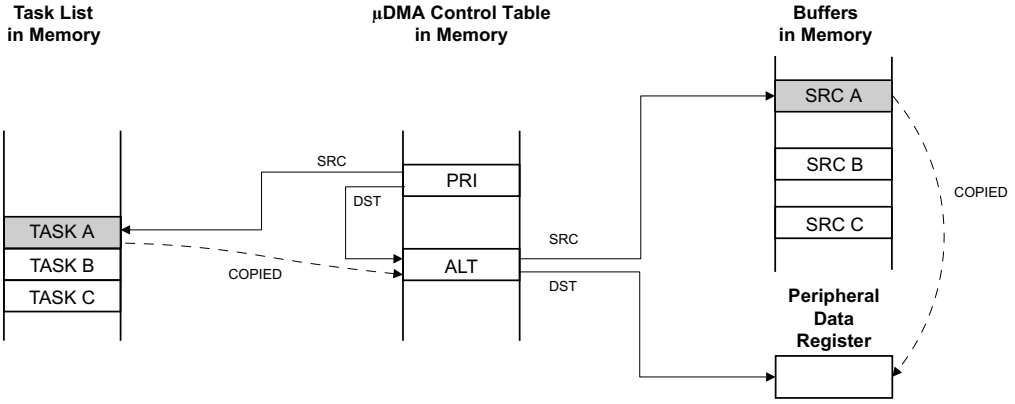
Figure 16-5. Peripheral Scatter-Gather, Setup and Configuration



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 "tasklist" 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.

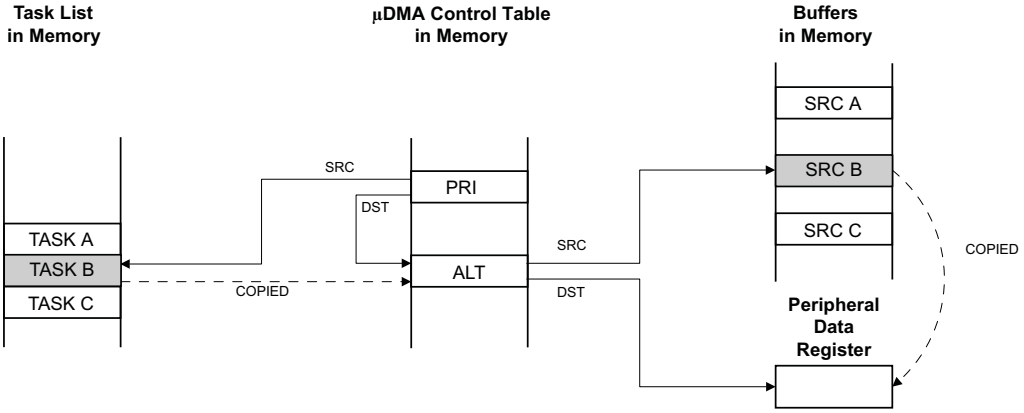
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-6. Peripheral Scatter-Gather, µDMA Copy Sequence



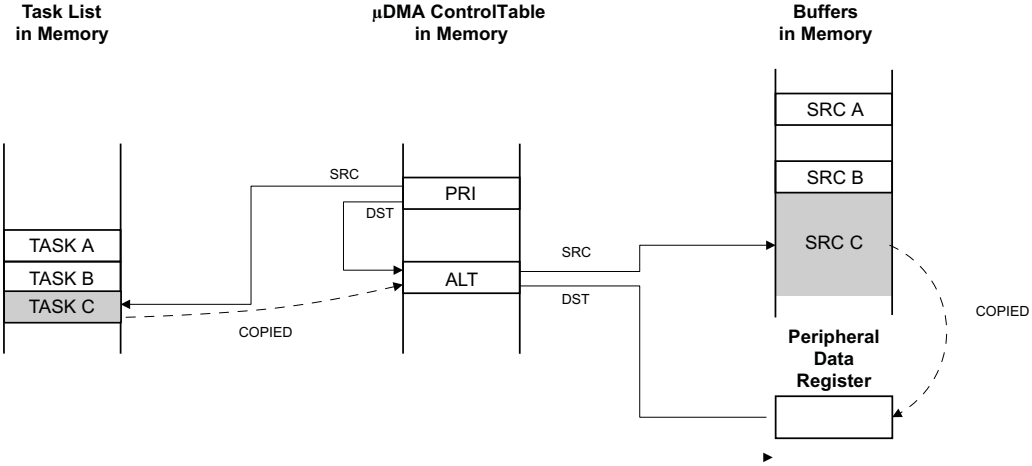
Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

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

Copyright © 2012–2019, Texas Instruments Incorporated

16.3.7 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 16-5 shows the configuration to read from a peripheral that supplies 8-bit data.

Table 16-5. 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

16.3.8 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 16-2). 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 Section 16.3.10 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.

16.3.9 Software Request

One µDMA channel is dedicated to software-initiated transfers. This channel also has a dedicated interrupt to signal completion of a µDMA transfer. A transfer is initiated by software by first configuring and enabling the transfer, and then issuing a software request using the DMA Channel Software Request (DMASWREQ) register. For software-based transfers, the Auto transfer mode should be used.

It is possible to initiate a transfer on any channel using the DMASWREQ register. If a request is initiated by software using a peripheral µDMA channel, then the completion interrupt occurs on the interrupt vector for the peripheral instead of the software interrupt vector. Any channel may be used for software requests as long as the corresponding peripheral is not using µDMA for data transfer.

Copyright © 2012–2019, Texas Instruments Incorporated

16.3.10 Interrupts and Errors

When a µDMA transfer is complete, the µDMA controller generates a completion interrupt on the interrupt vector of the peripheral. Therefore, if µDMA is used to transfer data for a peripheral and interrupts are used, then the interrupt handler for that peripheral must be designed to handle the µDMA transfer completion interrupt. If the transfer uses the software µDMA channel, then the completion interrupt occurs on the dedicated software µDMA interrupt vector (see Table 16-6).

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.

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 16-6 shows the dedicated interrupt assignments for the µDMA controller.

Table 16-6. µDMA Interrupt Assignments

Interrupt	Assignment
46	µDMA Software Channel Transfer
47	µDMA Error

16.4 Initialization and Configuration

16.4.1 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 peripheral in the System Control block. To do this, set the UDMA bit of the Run Mode Clock Gating Control Register 2 (RCGC2) register.
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.

16.4.2 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.

16.4.2.1 Configure the Channel Attributes

Follow these steps to 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

Copyright © 2012–2019, Texas Instruments Incorporated

controller to recognize requests for this channel.

16.4.2.2 Configure the Channel Control Structure

After configuring the channel attributes, 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 16-7.

Table 16-7. 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

16.4.2.2.1 Configure the Source and Destination

Now set the source and destination end pointers 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 16-8.

Table 16-8. 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

16.4.2.3 Start the Transfer

The channel is configured and is now 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.

16.4.3 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.

Copyright © 2012–2019, Texas Instruments Incorporated

16.4.3.1 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.

16.4.3.2 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 16-9.

Table 16-9. 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

16.4.3.2.1 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.

- Program the source end pointer at offset 0x070 to the address of the source buffer + 0x3F.
- 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 16-10.

Table 16-10. 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

Copyright © 2012–2019, Texas Instruments Incorporated

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.

16.4.3.3 Start the Transfer

Now the channel is configured and is ready to start.

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.

16.4.4 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.

16.4.4.1 Configure the Channel Attributes

First, configure the channel attributes:

- 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.
- Set bit 8 of the DMA Channel Primary Alternate Clear (DMAALTCLR) register to select the primary channel control structure for this transfer.
- Set bit 8 of the DMA Channel Useburst Clear (DMAUSEBURSTCLR) register to allow the µDMA controller to respond to single and burst requests.
- Set bit 8 of the DMA Channel Request Mask Clear (DMAREQMASKCLR) register to allow the µDMA controller to recognize requests for this channel.

16.4.4.2 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 16-11.

Table 16-11. 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

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-11. Primary and Alternate Channel Control Structure Offsets for Channel 8 (continued)

Offset	Description
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

16.4.4.2.1 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.

- Program the primary source end pointer at offset 0x080 to the address of the peripheral's receive buffer.
- Program the primary destination end pointer at offset 0x084 to the address of ping-pong buffer A + 0x3F.
- Program the alternate source end pointer at offset 0x280 to the address of the peripheral's receive buffer.
- 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.

- Program the primary channel control word at offset 0x088 according to Table 16-12.
- Program the alternate channel control word at offset 0x288 according to Table 16-12.

Table 16-12. 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.

Copyright © 2012–2019, Texas Instruments Incorporated

16.4.4.3 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.

- Configure and enable an interrupt handler for the peripheral.

16.4.4.4 Enable the µDMA Channel

Now the channel is configured and is ready to start.

Enable the channel by setting bit 8 of the DMA Channel Enable Set (DMAENASET) register.

16.4.4.5 Process Interrupts

The µDMA controller is now configured and enabled for transfer on channel 8. When the peripheral asserts the µDMA request signal, the µDMA controller makes transfers into buffer A using the primary channel control structure. When the primary transfer to buffer A is complete, it switches to the alternate channel control structure and makes transfers into buffer B. At the same time, the primary channel control word mode field is configured to indicate Stopped, and an interrupt is generated in the peripheral's raw interrupt status register.

When an interrupt is triggered, the interrupt handler must determine which buffer is complete and process the data or set a flag that the data must be processed by non-interrupt buffer processing code. Then the next buffer transfer must be set up.

In the interrupt handler:

- 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:
  - Process the newly received data in buffer A or signal the buffer processing code that buffer A has data available.
  - Reprogram the primary channel control word at offset 0x88 according to Table 16-12.
- 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:
  - Process the newly received data in buffer B or signal the buffer processing code that buffer B has data available.
  - Reprogram the alternate channel control word at offset 0x288 according to Table 16-12.

16.5 Register Map

lists the µDMA channel control structures and registers. The channel control structure shows the layout of one entry in the channel control table. The channel control table is located in system memory, and the location is determined by the application, that is, the base address is n/a (not applicable). In the table below, the offset for the channel control structures is the offset from the entry in the channel control table. See Section 16.3.5 and Table 16-3 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 (ending address of 0x400F.FFFF). Note that the µDMA module clock must be enabled before the registers can be programmed. There must be a delay of 3 system clocks after the µDMA module clock is enabled before any µDMA module registers are accessed.

Table 16-13.

Offset Name	Name	Type	Reset	Description
µDMA Channel Control Structure (Offset from Channel Control Table Base)				
0x000	DMASRCENDP	R/W	-	DMA Channel Source Address End Pointer
0x004	DMADSTENDP	R/W	-	DMA Channel Destination Address End Pointer
0x008	DMACHCTL	R/W	-	DMA Channel Control Word

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-13. (continued)

µDMA Registers (Offset from µDMA Base Address)				
0x000	DMASTAT	RO	0x001F.0000	DMA Status
0x004	DMACFG	WO		DMA Configuration
0x008	DMACTLBASE	R/W	0x0000.0000	DMA Channel Control Base Pointer
0x00C	DMAALTBASE	RO	0x0000.0200	DMA Alternate Channel Control Base Pointer
0x010	DMAWAITSTAT	RO	0xFFFF.FFC0	DMA Channel Wait-on-Request Status
0x014	DMASWREQ	WO	-	DMA Channel Software Request
0x018	DMAUSEBURSTSET	R/W	0x0000.0000	DMA Channel Useburst Set
0x01C	DMAUSEBURSTCLR	WO	-	DMA Channel Useburst Clear
0x020	DMAREQMASKSET	R/W	0x0000.0000	DMA Channel Request Mask Set
0x024	DMAREQMASKCLR	WO	-	DMA Channel Request Mask Clear
0x028	DMAENASET	R/W	0x0000.0000	DMA Channel Enable Set
0x02C	DMAENACLR-	WO	-	DMA Channel Enable Clear
0x030	DMAALTSET	R/W	0x0000.0000	DMA Channel Primary Alternate Set
0x034	DMAALTCLR	WO	-	DMA Channel Primary Alternate Clear
0x038	DMAPRIOSET	R/W	0x0000.0000	DMA Channel Priority Set
0x03C	DMAPRIOCLR	WO	-	DMA Channel Priority Clear 385
0x04C	DMAERRCLR	R/W	0x0000.0000	DMA Bus Error Clear
0x500	DMACHALT	R/W	0x0000.0000	DMA Channel Assignment
0x510	DMACHMAP0	R/W		DMA Channel Map Assignment 0
0x514	DMACHMAP1	R/W		DMA Channel Map Assignment 1
0x518	DMACHMAP2	R/W		DMA Channel Map Assignment 2
0x51C	DMACHMAP3	R/W		DMA Channel Map Assignment 3
0xFD0	DMAPeriphID4	RO	0x0000.0004	DMA Peripheral Identification 4
0xFE0	DMAPeriphID0	RO	0x0000.0030	DMA Peripheral Identification 0
0xFE4	DMAPeriphID1	RO	0x0000.00B2	DMA Peripheral Identification 1
0xFE8	DMAPeriphID2	RO	0x0000.000B	DMA Peripheral Identification 2
0xFEC	DMAPeriphID3	RO	0x0000.0000	DMA Peripheral Identification 3
0xFF0	DMAPCellID0	RO	0x0000.000D	DMA PrimeCell Identification 0
0xFF4	DMAPCellID1	RO	0x0000.00F0	DMA PrimeCell Identification 1
0xFF8	DMAPCellID2	RO	0x0000.0005	DMA PrimeCell Identification 2
0xFFC	DMAPCellID3	RO	0x0000.00B1	DMA PrimeCell Identification 3

Copyright © 2012–2019, Texas Instruments Incorporated

16.6 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 Section 16.3.5 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.

16.6.1 DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000

DMA Channel Source Address End Pointer (DMASRCENDP) is part of the Channel Control Structure and is used to specify the source address for a µDMA transfer.

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

NOTE: The offset specified is from the base address of the control structure in system memory, not the µDMA module base address.

Figure 16-7. DMA Channel Source Address End Pointer (DMASRCENDP) Register

31	0
ADDR	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-14. DMA Channel Source Address End Pointer (DMASRCENDP) Register Field Descriptions

Bit	Field	Value	Description
31-0	ADDR		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).

16.6.2 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.

Figure 16-8. DMA Channel Destination Address End Pointer (DMADSTENDP) Register

31	0
ADDR	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-15. DMA Channel Destination Address End Pointer (DMADSTENDP) Register Field Descriptions

Bit	Field	Value	Description
31-0	ADDR		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).

16.6.3 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.

Figure 16-9. DMA Channel Control Word (DMACHCTL) Register

31	30	29	28	27	26	25	24
DSTINC		DSTSIZE		SRCINC		SRCSIZE	
R/W		R/W		R/W		R/W	

23	18	17	16
Reserved		ARBSIZE	
R/W		R/W	

15	14	13	8
ARBSIZE		XFERSIZE	
R/W		R/W	

7	4	3	2	0
XFERSIZE		NXTUSEBURST	XFERMODE	
R/W		R/W	R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-16. DMA Channel Control Word (DMACHCTL) Register Field Descriptions

Bit	Field	Value	Description
31-30	DSTINC		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).
		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

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-16. DMA Channel Control Word (DMACHCTL) Register Field Descriptions (continued)

Bit	Field	Value	Description
29-28	DSTSIZE		Destination Data Size This field configures the destination item data size. **Note:** DSTSIZE must be the same as SRCSIZE.
		0x0	Byte 8-bit data size
		0x1	Half-word 16-bit data size
		0x2	Word 32-bit data size
		0x3	Reserved
27-26	SRCINC		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).
		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		Source Data Size This field configures the source item data size. **Note:** DSTSIZE must be the same as SRCSIZE.
		0x0	Byte 8-bit data size.
		0x1	Half-word 16-bit data size.
		0x2	Word 32-bit data size.
		0x3	Reserved
23-18	Reserved		Reserved
17-14	ARBSIZE		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.
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-16. DMA Channel Control Word (DMACHCTL) Register Field Descriptions (continued)

Bit	Field	Value	Description
13-4	XFERSIZE		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		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.
2-0	XFERMODE		µDMA Transfer Mode This field configures the operating mode of the µDMA cycle. Refer to Section 16.3.6 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.
		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 Section 16.3.6.4.

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 Â 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 Section 16.3.6.5.

Copyright © 2012–2019, Texas Instruments Incorporated

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 Section 16.3.6.6.

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.

16.7 µDMA Register Descriptions

The register addresses given are relative to the µDMA base address of 0x400F.F000.

16.7.1 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.

Figure 16-10. DMA Status (DMASTAT) Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											DMACHANS				
R-0											R-1F				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								STATE				Reserved			MASTEN
R-0								R-0				R-0			R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-17. DMA Status (DMASTAT) Register Field Descriptions

Bit	Field	Value	Description
31-21	Reserved		Reserved
20-16	DMACHANS		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		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-17. DMA Status (DMASTAT) Register Field Descriptions (continued)

Bit	Field	Value	Description
7-4	STATE		Control State Machine Status This field shows the current status of the control state machine. Status can be one of the following.
		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
3-1	Reserved		Reserved
0	MASTEN		Master Enable Status
		0	The µDMA controller is disabled.
		1	The µDMA controller is enabled.

16.7.2 DMA Configuration (DMACFG), offset 0x004

The DMACFG register controls the configuration of the µDMA controller.

Figure 16-11. DMA Configuration (DMACFG) Register

31	1	0
Reserved		MASTEN
W		W

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-18. DMA Configuration (DMACFG) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	MASTEN		Controller Master Enable
		0	Disables the µDMA controller.
		1	Enables µDMA controller.

16.7.3 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 Section 16.3.5 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.

Figure 16-12. DMA Channel Control Base Pointer (DMACTLBASE) Register

31	10	9	0
ADDR		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-19. DMA Channel Control Base Pointer (DMACTLBASE) Register Field Descriptions

Bit	Field	Value	Description
31-10	ADDR		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		Reserved

16.7.4 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.

Figure 16-13. DMA Alternate Channel Control Base Pointer (DMAALTBASE) Register

31	0
ADDR	
R-0000.0200	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-20. DMA Alternate Channel Control Base Pointer (DMAALTBASE) Register Field Descriptions

Bit	Field	Value	Description
31-0	ADDR		Alternate Channel Address Pointer This field provides the base address of the alternate channel control structures.

16.7.5 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.

Figure 16-14. DMA Channel Wait-on-Request Status (DMAWAITSTAT) Register

31	0
WAITREQ[n]	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-21. DMA Channel Wait-on-Request Status (DMAWAITSTAT) Register Field Descriptions

Bit	Field	Value	Description
31-0	WAITREQ[n]		Channel [n] Wait Status These bits provide the channel wait-on-request status. Bit 0 corresponds to channel 0.
		0	The corresponding channel is not waiting on a request.
		1	The corresponding channel is waiting on a request.

16.7.6 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.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-15. DMA Channel Software Request (DMASWREQ) Register

31	0
SWREQ[n]	
W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-22. DMA Channel Software Request (DMASWREQ) Register Field Descriptions

Bit	Field	Value	Description
31-0	SWREQ[n]		Channel [n] Software Request These bits generate software requests. Bit 0 corresponds to channel 0. These bits are automatically cleared when the software request has been completed.
		0	No request generated.
		1	Generate a software request for the corresponding channel.

16.7.7 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 Section 16.3.4 for more details about request types.

Figure 16-16. DMA Channel Useburst Set (DMAUSEBURSTSET) Register

31	0
SET[n]	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-23. DMA Channel Useburst Set (DMAUSEBURSTSET) Register Field Descriptions

Bit	Field	Value	Description
31-0	SET[n]		Channel [n] Useburst Set 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.
		0	µDMA channel [n] responds to single or burst requests.
		1	µDMA channel [n] responds only to burst requests.

16.7.8 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.

Figure 16-17. DMA Channel Useburst Clear (DMAUSEBURSTCLR) Register

31	0
CLR[n]	
W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-24. DMA Channel Useburst Clear (DMAUSEBURSTCLR) Register Field Descriptions

Bit	Field	Value	Description
31-0	CLR[n]		Channel [n] Useburst Clear
		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.

16.7.9 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.

Figure 16-18. DMA Channel Request Mask Set (DMAREQMASKSET) Register

31	0
SET[n]	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-25. DMA Channel Request Mask Set (DMAREQMASKSET) Register Field Descriptions

Bit	Field	Value	Description
31-0	SET[n]		Channel [n] Request Mask Set Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the DMAREQMASKCLR register.
		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.

16.7.10 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.

Figure 16-19. DMA Channel Request Mask Clear (DMAREQMASKCLR) Register

31	0
CLR[n]	
W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-26. DMA Channel Request Mask Clear (DMAREQMASKCLR) Register Field Descriptions

Bit	Field	Value	Description
31-0	CLR[n]		Channel [n] Request Mask Clear
		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.

16.7.11 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.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-20. DMA Channel Enable Set (DMAENASET) Register

31	0
SET[n]	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-27. DMA Channel Enable Set (DMAENASET) Register Field Descriptions

Bit	Field	Value	Description
31-0	SET[n]		Channel [n] Enable Set Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the DMAENACLR register.
		0	µDMA Channel [n] is disabled.
		1	µDMA Channel [n] is enabled.

16.7.12 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.

Figure 16-21. DMA Channel Enable Clear (DMAENACLR) Register

31	0
CLR[n]	
W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-28. DMA Channel Enable Clear (DMAENACLR) Register Field Descriptions

Bit	Field	Value	Description
31-0	CLR[n]		Clear Channel [n] Enable Clear **Note:** The controller disables a channel when it completes the µDMA cycle.
		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.

16.7.13 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.

Figure 16-22. DMA Channel Primary Alternate Set (DMAALTSET) Register

31	0
SET[n]	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-29. DMA Channel Primary Alternate Set (DMAALTSET) Register Field Descriptions

Bit	Field	Value	Description
31-0	SET[n]		Channel [n] Alternate Set 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.
		0	µDMA channel [n] is using the primary control structure.
		1	µDMA channel [n] is using the alternate control structure.

Copyright © 2012–2019, Texas Instruments Incorporated

16.7.14 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.

Figure 16-23. DMA Channel Primary Alternate Clear (DMAALTCLR) Register

31	0
CLR[n]	
W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-30. DMA Channel Primary Alternate Clear (DMAALTCLR) Register Field Descriptions

Bit	Field	Value	Description
31-0	CLR[n]		Channel [n] Alternate Clear **Note:** For Ping-Pong and Scatter-Gather cycle types, the µDMA controller automatically sets these bits to select the alternate channel control data structure.
		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.

16.7.15 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.

Figure 16-24. DMA Channel Priority Set (DMAPRIOSET) Register

31	0
SET[n]	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-31. DMA Channel Priority Set (DMAPRIOSET) Register Field Descriptions

Bit	Field	Value	Description
31-0	SET[n]		Channel [n] Priority Set Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the DMAPRIOCLR register.
		0	µDMA channel [n] is using the default priority level.
		1	µDMA channel [n] is using a high priority level.

16.7.16 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.

Figure 16-25. DMA Channel Priority Clear (DMAPRIOCLR) Register

31	0
CLR[n]	
W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-32. DMA Channel Priority Clear (DMAPRIOCLR) Register Field Descriptions

Bit	Field	Value	Description
31-0	CLR[n]		Channel [n] Priority Clear
		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.

16.7.17 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.

Figure 16-26. DMA Bus Error Clear (DMAERRCLR) Register

31	1	0
Reserved		ERRCLR
R-0		R/W/1C-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-33. DMA Bus Error Clear (DMAERRCLR) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	ERRCLR		µDMA Bus Error Status This bit is cleared by writing a 1 to it.
		0	No bus error is pending.
		1	A bus error is pending.

16.7.18 DMA Channel Assignment (DMACHALT), offset 0x500

Each bit of the DMACHALT register represents the corresponding µDMA channel. Setting a bit selects the secondary channel assignment as specified in Table 16-1.

Figure 16-27. DMA Channel Assignment (DMACHALT) Register

31	0
CHASGN[n]	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-34. DMA Channel Assignment (DMACHALT) Register Field Descriptions

Bit	Field	Value	Description
31-0	CHASGN[n]		Channel [n] Assignment Select
		0	Use the primary channel assignment.
		1	Use the secondary channel assignment.

16.7.19 DMA Channel Map Assignment (DMACHMAP0) Register, offset 0x510

Each bit of the DMACHMAP0 register controls the channel assignments for the first, second, and third mapping.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-28. DMA Channel Map Assignment (DMACHMAP0) Register

31	0
CHMAP0	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-35. DMA Channel Map Assignment (DMACHMAP0) Register Field Descriptions

Bit	Field	Value	Description
31-28		0	Channel 7 First Assignment
		1	Channel 7 Second Assignment
		2	Channel 7 Third Assignment
		3	Reserved
27-24		0	Channel 6 First Assignment
		1	Channel 6 Second Assignment
		2	Channel 6 Third Assignment
		3	Reserved
23-20		0	Channel 5 First Assignment
		1	Channel 5 Second Assignment
		2	Channel 5 Third Assignment
		3	Reserved
19-16		0	Channel 4 First Assignment
		1	Channel 4 Second Assignment
		2	Channel 4 Third Assignment
		3	Reserved
15-12		0	Channel 3 First Assignment
		1	Channel 3 Second Assignment
		2	Channel 3 Third Assignment
		3	Reserved
11-8		0	Channel 2 First Assignment
		1	Channel 2 Second Assignment
		2	Channel 2 Third Assignment
		3	Reserved
7-4		0	Channel 1 First Assignment
		1	Channel 1 Second Assignment
		2	Channel 1 Third Assignment
		3	Reserved
3-0		0	Channel 0 First Assignment
		1	Channel 0 Second Assignment
		2	Channel 0 Third Assignment
		3	Reserved

16.7.20 DMA Channel Map Assignment (DMACHMAP1) Register, offset 0x514

Each bit of the DMACHMAP0 register controls the channel assignments for the first, second, and third mapping.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-29. DMA Channel Map Assignment (DMACHMAP1) Register

31	0
CHMAP1	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-36. DMA Channel Map Assignment (DMACHMAP1) Register Field Descriptions

Bit	Field	Value	Description
31-28		0	Channel 15 First Assignment
		1	Channel 15 Second Assignment
		2	Channel 15 Third Assignment
		3	Reserved
27-24		0	Channel 14 First Assignment
		1	Channel 14 Second Assignment
		2	Channel 14 Third Assignment
		3	Reserved
23-20		0	Channel 13 First Assignment
		1	Channel 13 Second Assignment
		2	Channel 13 Third Assignment
		3	Reserved
19-16		0	Channel 12 First Assignment
		1	Channel 12 Second Assignment
		2	Channel 12 Third Assignment
		3	Reserved
15-12		0	Channel 11 First Assignment
		1	Channel 11 Second Assignment
		2	Channel 11 Third Assignment
		3	Reserved
11-8		0	Channel 10 First Assignment
		1	Channel 10 Second Assignment
		2	Channel 10 Third Assignment
		3	Reserved
7-4		0	Channel 9 First Assignment
		1	Channel 9 Second Assignment
		2	Channel 9 Third Assignment
		3	Reserved
3-0		0	Channel 8 First Assignment
		1	Channel 8 Second Assignment
		2	Channel 8 Third Assignment
		3	Reserved

16.7.21 DMA Channel Map Assignment (DMACHMAP2) Register, offset 0x518

Each bit of the DMACHMAP0 register controls the channel assignments for the first, second, and third mapping.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-30. DMA Channel Map Assignment (DMACHMAP2) Register

31	0
CHMAP2	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-37. DMA Channel Map Assignment (DMACHMAP2) Register Field Descriptions

Bit	Field	Value	Description
31-28		0	Channel 23 First Assignment
		1	Channel 23 Second Assignment
		2	Channel 23 Third Assignment
		3	Reserved
27-24		0	Channel 22 First Assignment
		1	Channel 22 Second Assignment
		2	Channel 22 Third Assignment
		3	Reserved
23-20		0	Channel 21 First Assignment
		1	Channel 21 Second Assignment
		2	Channel 21 Third Assignment
		3	Reserved
19-16		0	Channel 20 First Assignment
		1	Channel 20 Second Assignment
		2	Channel 20 Third Assignment
		3	Reserved
15-12		0	Channel 19 First Assignment
		1	Channel 19 Second Assignment
		2	Channel 19 Third Assignment
		3	Reserved
11-8		0	Channel 18 First Assignment
		1	Channel 18 Second Assignment
		2	Channel 18 Third Assignment
		3	Reserved
7-4		0	Channel 17 First Assignment
		1	Channel 17 Second Assignment
		2	Channel 17 Third Assignment
		3	Reserved
3-0		0	Channel 16 First Assignment
		1	Channel 16 Second Assignment
		2	Channel 16 Third Assignment
		3	Reserved

16.7.22 DMA Channel Map Assignment (DMACHMAP3) Register, offset 0x51C

Each bit of the DMACHMAP0 register controls the channel assignments for the first, second, and third mapping.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-31. DMA Channel Map Assignment (DMACHMAP3) Register

31	0
CHMAP3	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-38. DMA Channel Map Assignment (DMACHMAP3) Register Field Descriptions

Bit	Field	Value	Description
31-28		0	Channel 31 First Assignment
		1	Channel 31 Second Assignment
		2	Channel 31 Third Assignment
		3	Reserved
27-24		0	Channel 30 First Assignment
		1	Channel 30 Second Assignment
		2	Channel 30 Third Assignment
		3	Reserved
23-20		0	Channel 29 First Assignment
		1	Channel 29 Second Assignment
		2	Channel 29 Third Assignment
		3	Reserved
19-16		0	Channel 28 First Assignment
		1	Channel 28 Second Assignment
		2	Channel 28 Third Assignment
		3	Reserved
15-12		0	Channel 27 First Assignment
		1	Channel 27 Second Assignment
		2	Channel 27 Third Assignment
		3	Reserved
11-8		0	Channel 26 First Assignment
		1	Channel 26 Second Assignment
		2	Channel 26 Third Assignment
		3	Reserved
7-4		0	Channel 25 First Assignment
		1	Channel 25 Second Assignment
		2	Channel 25 Third Assignment
		3	Reserved
3-0		0	Channel 24 First Assignment
		1	Channel 24 Second Assignment
		2	Channel 24 Third Assignment
		3	Reserved

16.7.23 DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0

The DMAPeriphIDn registers are hard-coded, and the fields within the registers determine the reset values.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 16-32. DMA Peripheral Identification 0 (DMAPeriphID0) Register

31	8	7	0
Reserved		PID0	
R-0		R-30	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

16.7.24 DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4

The DMAPeriphIDn registers are hard-coded, and the fields within the registers determine the reset values.

Figure 16-33. DMA Peripheral Identification 1 (DMAPeriphID1) Register

31	8	7	0
Reserved		PID1	
R-0		R-B2	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-39. DMA Peripheral Identification 1 (DMAPeriphID1) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID1		µDMA Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

16.7.25 DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8

The DMAPeriphIDn registers are hard-coded, and the fields within the registers determine the reset values.

Figure 16-34. DMA Peripheral Identification 2 (DMAPeriphID2) Register

31	8	7	0
Reserved		PID2	
R-0		R-0B	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-40. DMA Peripheral Identification 2 (DMAPeriphID2) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID2		µDMA Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

16.7.26 DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC

The DMAPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 16-35. DMA Peripheral Identification 3 (DMAPeriphID3) Register

31	8	7	0
Reserved		PID3	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-41. DMA Peripheral Identification 3 (DMAPeriphID3) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID3		µDMA Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

16.7.27 DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0

The DMAPeriphIDn registers are hard-coded, and the fields within the registers determine the reset values.

Figure 16-36. DMA Peripheral Identification 4 (DMAPeriphID4) Register

31	8	7	0
Reserved		PID4	
R-0		R-04	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-42. DMA Peripheral Identification 4 (DMAPeriphID4) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID4		µDMA Peripheral ID Register Can be used by software to identify the presence of this peripheral.

16.7.28 DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0

The DMAPCellIDn registers are hard-coded, and the fields within the registers determine the reset values.

Figure 16-37. DMA PrimeCell Identification 0 (DMAPCellID0) Register

31	8	7	0
Reserved		CID0	
R-0		R-0D	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-43. DMA PrimeCell Identification 0 (DMAPCellID0) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID0		µDMA PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

16.7.29 DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4

The DMAPCellIDn registers are hard-coded, and the fields within the registers determine the reset values.

Figure 16-38. DMA PrimeCell Identification 1 (DMAPCellID1) Register

31	8	7	0
Reserved		CID1	
R-0		R-F0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 16-44. DMA PrimeCell Identification 1 (DMAPCellID1) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID1		µDMA PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

16.7.30 DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8

The DMAPCellIDn registers are hard-coded, and the fields within the registers determine the reset values.

Figure 16-39. DMA PrimeCell Identification 2 (DMAPCellID2) Register

31	8	7	0
Reserved		CID2	
R-0		R-05	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-45. DMA PrimeCell Identification 2 (DMAPCellID2) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID2		µDMA PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

16.7.31 DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC

The DMAPCellIDn registers are hard-coded, and the fields within the registers determine the reset values.

Figure 16-40. DMA PrimeCell Identification 3 (DMAPCellID3) Register

31	8	7	0
Reserved		CID3	
R-0		R-B1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-46. DMA PrimeCell Identification 3 (DMAPCellID3) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID3		µDMA PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

Copyright © 2012–2019, Texas Instruments Incorporated

External Peripheral Interface (EPI)

This chapter discusses the function of the external peripheral interface (EPI).

Topic **Page**

17.1 Introduction 1191
17.2 EPI Block Diagram 1192
17.3 Functional Description 1192
17.4 DMA Operation 1194
17.5 Initialization and Configuration 1194
17.6 SDRAM Mode 1195
17.7 Host Bus Mode 1199
17.8 General-Purpose Mode 1215
17.9 C28x Access to EPI 1223
17.10 Register Map 1225
17.11 Register Descriptions 1228

Copyright © 2012–2019, Texas Instruments Incorporated

17.1 Introduction

The external peripheral interface (EPI) is a high-speed parallel bus for external peripherals or memory. It has several modes of operation to interface gluelessly to many types of external devices. The EPI is similar to a standard microprocessor address/data bus, except that it must typically be connected to just one type of external device. Enhanced capabilities include µDMA support, clocking control and support for external FIFO buffers.

The EPI has the following features:

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

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

- Synchronous Dynamic Random Access Memory (SDRAM)
  - Supports x16 (single data rate) SDRAM at up to 62.5 MHz
  - Supports low-cost SDRAMs up to 64 MB (512 megabits)
  - Includes automatic refresh and access to all banks/rows
  - Includes a Sleep/Standby mode to keep contents active with minimal power draw
  - Multiplexed address/data interface for reduced pin count
- Host-bus
  - Traditional x8 and x16 MCU bus interface capabilities
  - Similar device compatibility options as PIC, ATmega, 8051, and others
  - Access to SRAM, NOR Flash memory, and other devices, with up to 1 MB of addressing in non-multiplexed mode and 256 MB in multiplexed mode (512 MB in Host-Bus 16 mode with no byte selects)
  - Support of both muxed and de-muxed address and data
  - Access to a range of devices supporting the non-address FIFO x8 and x16 interface variant, with support for external FIFO (XFIFO) EMPTY and FULL signals
  - Speed controlled, with read and write data wait-state counters
  - Support for read/write burst mode to Host Bus
  - Multiple chip select modes including single, dual, and quad chip selects, with and without ALE
  - External iRDY signal provided for stall capability of reads and writes
  - Manual chip-enable (or use extra address pins)
- General Purpose
  - Wide parallel interfaces for fast communications with CPLDs and FPGAs
  - Data widths up to 32 bits
  - Data rates up to 150 MB/second
  - Optional "address" sizes from 4 bits to 20 bits
  - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input
- General parallel GPIO

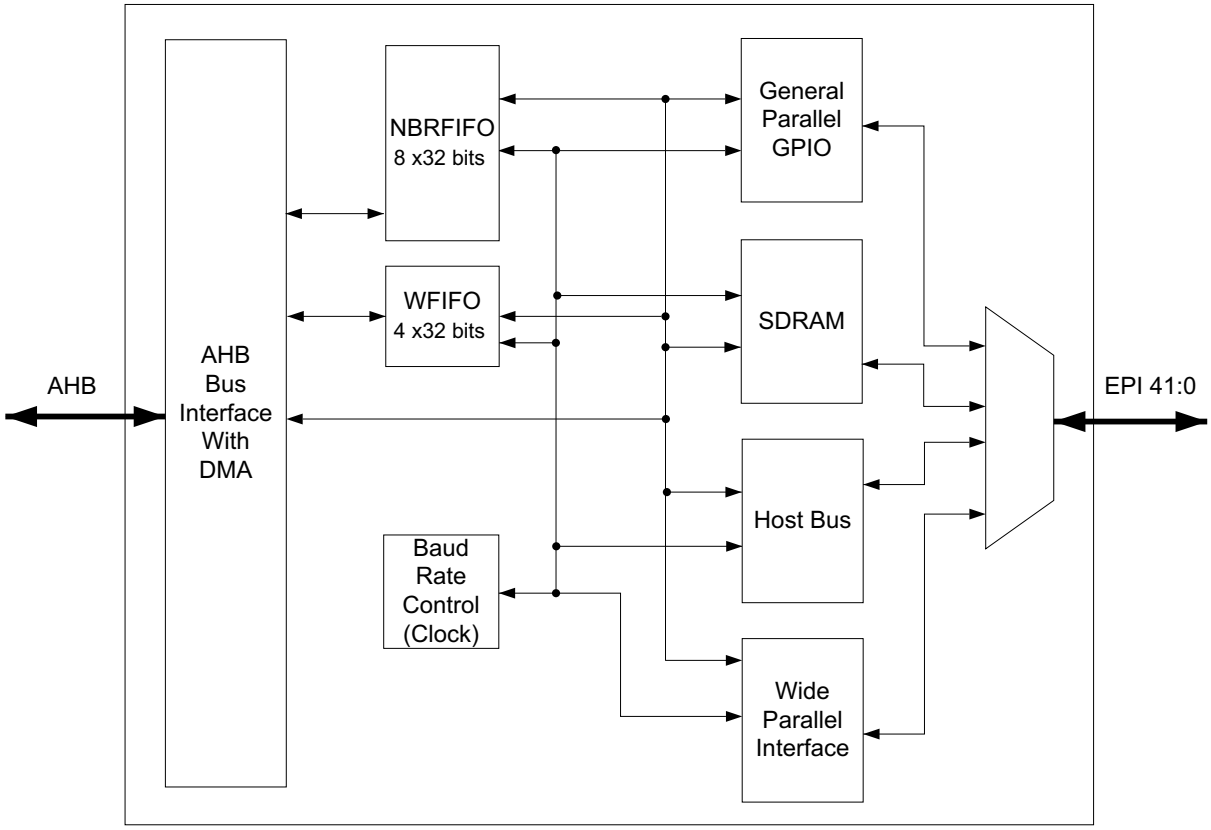
Copyright © 2012–2019, Texas Instruments Incorporated

– 1 to 32 bits, FIFOed with speed control
– Useful for custom peripherals or for digital data acquisition and actuator controls

17.2 EPI Block Diagram

Figure 17-1 provides a block diagram of the EPI module.

Figure 17-1. EPI Block Diagram



17.3 Functional Description

The EPI controller provides a glueless, programmable interface to a variety of common external peripherals such as SDRAM x16, Host Bus x8 and x16 devices, RAM, NOR Flash memory, CPLDs and FPGAs. In addition, the EPI controller provides custom GPIO that can use a FIFO with speed control by using either the internal write FIFO (WFIFO) or the non-blocking read FIFO (NBRFIFO). The WFIFO can hold four words of data that are written to the external interface at the rate controlled by the EPI Main Baud Rate (EPIBAUD) registers. The NBRFIFO can hold eight words of data and samples at the rate controlled by the EPIBAUD register. The EPI controller provides predictable operation and thus has an advantage over regular GPIO which has more variable timing due to on-chip bus arbitration and delays across bus bridges. Blocking reads stall the CPU until the transaction completes. Non-blocking reads are performed in the background and allow the processor to continue operation. In addition, write data can also be stored in the WFIFO to allow multiple writes with no stalls.

NOTE: Both the WTAV bit field in the EPIWFIFOCNT register and the WBUSY bit in the EPISTAT register must be polled to determine if there is a current write transaction from the WFIFO. If both of these bits are clear, then a new bus access may begin.

Copyright © 2012–2019, Texas Instruments Incorporated

Main read and write operations can be performed in subsets of the range 0x6000.0000 to 0xDFFF.FFFF. A read from an address mapped location uses the offset and size to control the address and size of the external operation. When performing a multi-value load, the read is done as a burst (when available)to maximize performance. A write to an address mapped location uses the offset and size to control the address and size of the external operation. When performing a multi-value store, the write is done as a burst (when available) to maximize performance.

NAND Flash memory (x8) can be read natively. Automatic programming support is not provided; programming must be done by the user following the manufacturer's protocol. Automatic page ECC is also not supported, but can be performed in software.

17.3.1 Non-Blocking Reads

The EPI Controller supports a special kind of read called a non-blocking read, also referred to as a posted read. Where a normal read stalls the processor or µDMA until the data is returned, a non-blocking read is performed in the background.

A non-blocking read is configured by writing the start address into a EPIRADDRn register, the size per transaction into a EPIRSIZEn register, and then the count of operations into a EPIRPSTDn register. After each read is completed, the result is written into the NBRFIFO and the EPIRADDRn register is incremented by the size (1, 2, or 4). The three most significant bits of EPIRADDRn register are only relevant in the Host Bus multi-chip select mode when they are used to enable the different chip selects.

If the NBRFIFO is filled, then the reads pause until space is made available. The NBRFIFO can be configured to interrupt the processor or trigger the µDMA based on fullness using the EPIFIFOLVL register. By using the trigger/interrupt method, the µDMA (or processor)can keep space available in the NBRFIFO and allow the reads to continue unimpeded.

When performing non-blocking reads, the SDRAM controller issues two additional read transactions after the burst request is terminated. The data for these additional transfers is discarded. This situation is transparent to the user other than the additional EPI bus activity and can safely be ignored.

Two non-blocking read register sets are available to allow sequencing and ping-pong use. When one completes, the other then activates. So, for example, if 20 words are to be read from 0x100 and 10 words from 0x200, the EPIRPSTD0 register can be set up with the read from 0x100 (with a count of 20), and the EPIRPSTD1 register can be set up with the read from 0x200 (with a count of 10). When EPIRPSTD0 finishes (count goes to 0), the EPIRPSTD1 register then starts its operation. The NBRFIFO has then passed 30 values. When used with the µDMA, it may transfer 30 values (simple sequence), or the primary/alternate model may be used to handle the first 20 in one way and the second 10 in another. It is also possible to reload the EPIRPSTD0 register when it is finished (and the EPIRPSTD1 register is active); thereby, keeping the interface constantly busy.

To cancel a non-blocking read, the EPIRPSTDn register is cleared. Care must be taken, however, if the register set was active to drain away any values read into the NBRFIFO and ensure that any read in progress is allowed to complete.

To ensure that the cancel is complete, the following algorithm is used (using the EPIRPSTD0 register for example):

```
EPIRPSTD0 = 0;
while ((EPISTAT & 0x11) == 0x10)
; // we are active and busy
// if here, then other one is active or interface no longer busy
cnt = (EPIRADDR0 - original_address)/ EPIRSIZE0; // count of values read
cnt -= values_read_so_far;
// cnt is now number left in FIFO
while (cnt--)
value = EPIREADFIFO; // drain
```


Copyright © 2012–2019, Texas Instruments Incorporated

The above algorithm can be optimized in code; however, the important point is to wait for the cancel to complete because the external interface could have been in the process of reading a value when the cancel came in, and it must be allowed to complete.

17.4 DMA Operation

The µDMA can be used to achieve maximum transfer rates on the EPI through the NBRFIFO and the WFIFO. The µDMA has one channel for write and one for read. For writes, the EPI DMA Transmit Count (EPIDMATXCNT) register is programmed with the total number of transfers by the µDMA. An equivalent value is programmed into the DMA Channel Control Word (DMACHCTL) register of the uDMA at offset 0x008. A µDMA request is asserted by the EPI WRFIFO when the TXCNT value of the EPIDMATXCNT register is greater than zero and the WTAV bit field of the EPIWFIFOCNT register is less than the programmed threshold trigger, WRFIFO, of the EPIFIFOLVL register. The write channel continues to write data until the TXCNT value in the EPIDMACNT register is zero.

NOTE: When the WRFIFO bit in the EPIFIFOLVL register is set to 0x4 and the application bursts four words to an empty FIFO, the WRFIFO trigger may or may not de-assert depending on if all four words were written to the WRFIFO or if the first word was passed immediately to the function requiring it. Thus, the application may not see the WRRIS bit in the EPIRIS register clear on a burst of four words.

The non-blocking read channel copies values from the NBRFIFO when the NBRFIFO is at the level specified by the EPIFIFOLVL register. For non-blocking reads, the start address, the size per transaction, and the count of elements must be programmed in the µDMA. Note that both non-blocking read register sets can be used, and they fill the NBRFIFO such that one runs to completion, then the next one starts (they do not interleave). Using the NBRFIFO provides the best possible transfer rate.

For blocking reads, the µDMA software channel (or another unused channel) is used for memory-to-memory transfers (or memory to peripheral, where some other peripheral is used). In this situation, the µDMA stalls until the read is complete and is not able to service another channel until the read is done. As a result, the arbitration size should normally be programmed to one access at a time. The µDMA controller can also transfer from and to the NBRFIFO and the WFIFO using the µDMA software channel in memory mode, however, the µDMA is stalled once the NBRFIFO is empty or the WFIFO is full. Note that when the µDMA controller is stalled, the core continues operation. See the *Micro Direct Memory Access (µDMA)* chapter for more information on configuring the µDMA.

The size of the FIFOs must be taken into consideration when configuring the µDMA to transfer data to and from the EPI. The arbitration size should be four or less when writing to EPI address space and eight or less when reading from EPI address space.

17.5 Initialization and Configuration

To enable and initialize the EPI controller, the following steps are necessary:

- Enable the EPI module using the RCGC1 register (see the *System Control* chapter).
- Enable the clock to the appropriate GPIO module via the RCGC2 register. To find out which GPIO port to enable, see the *GPIO* chapter.
- Set the GPIO AFSEL bits for the appropriate pins. To determine which GPIOs to configure, see the *GPIO* chapter.
- Configure the GPIO current level and/or slew rate as specified for the mode selected. See the *GPIO* chapter.
- Configure the PMCn fields in the GPIOPCTL register to assign the EPI signals to the appropriate pins (see the *GPIO* chapter)
- Select the mode for the EPI block to SDRAM, HB8, HB16, or general parallel use, using the MODE field in the EPI Configuration (EPICFG) register. Set the mode-specific details (if needed) using the appropriate mode configuration EPI Host Bus Configuration (EPIHBnCFGn) registers for the desired chip-select configuration. Set the EPI Main Baud Rate (EPIBAUD) and EPI Main Baud Rate 2 (EPIBAUD2) register if the baud rate must be slower than the system clock rate.
- Configure the address mapping using the EPI Address Map (EPIADDRMAP) register. The selected

Copyright © 2012–2019, Texas Instruments Incorporated

start address and range is dependent on the type of external device and maximum address (as appropriate). For example, for a 512-megabit SDRAM, program the ERADR field to 0x1 for address 0x6000.0000 or 0x2 for address 0x8000.0000; and program the ERSZ field to 0x3 for 256 MB. If using General-Purpose mode and no address at all, program the EPADR field to 0x1 for address 0xA000.0000 or 0x2 for address 0xC000.0000; and program the EPSZ field to 0x0 for 256 bytes.

- To read or write directly, use the mapped address area (configured with the EPIADDRMAP register). Up to four or five writes can be performed at once without blocking. Each read is blocked until the value is retrieved.
- To perform a non-blocking read, see Section 17.6.4.

NOTE: The application should not attempt to access externally until eight system clock cycles after the EPI has been fully configured.

The following subsections describe the initialization and configuration for each of the modes of operation. Proper initialization must be done to ensure correct operation. Control of the GPIO states is also important, as changes may cause the external device to interpret pin states as actions or commands (see the *GPIO* chapter, Register Descriptions). Normally, a pull-up or pull-down is needed on the board to at least control the chip-select or chip-enable as the GPIOs come out of reset in tri-state.

17.6 SDRAM Mode

When activating the SDRAM mode, it is important to consider a few points:

- Generally, it takes over 100 μs from when the mode is activated to when the first operation is allowed. The SDRAM controller begins the SDRAM initialization sequence as soon as the mode is selected and enabled via the EPICFG register. It is important that the GPIOs are properly configured before the SDRAM mode is enabled, as the EPI controller is relying on the GPIO block's ability to drive the pins immediately. As part of the initialization sequence, the LOAD MODE REGISTER command is automatically sent to the SDRAM with a value of 0x27, which sets a $\overline{\text{CAS}}$ latency of 2 and a full page burst length.
- The INITSEQ bit in the EPI Status (EPISTAT) register can be checked to determine when the initialization sequence is complete.
- When using a frequency range and/or refresh value other than the default value, it is important to configure the FREQ and RFSH fields in the EPI SDRAM Configuration (EPISDRAMCFG) register shortly after activating the mode. After the 100-μs startup time, the EPI block must be configured properly to keep the SDRAM contents stable..
- The SLEEP bit in the EPISDRAMCFG register may be configured to put the SDRAM into a low-power self-refreshing state. It is important to note that the SDRAM mode must not be disabled once enabled, or else the SDRAM is no longer clocked and the contents are lost.

The SIZE field of the EPISDRAMCFG register must be configured correctly based on the amount of SDRAM in the system.

The FREQ field must be configured according to the value that represents the range being used. Based on the range selected, the number of external clocks used between certain operations (for example, PRECHARGE or ACTIVATE) is determined. If a higher frequency is given than is used, then the only downside is that the peripheral is slower (uses more cycles for these delays). If a lower frequency is given, incorrect operation occurs.

See your device-specific data manual's *EPI Electricals* section for timing details for the SDRAM mode.

Copyright © 2012–2019, Texas Instruments Incorporated

17.6.1 External Signal Connections

Table 17-1 defines how EPI module signals should be connected to SDRAMs. The table applies when using a x16 SDRAM up to 512 megabits. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Table 17-1. EPI SDRAM Signal Connections

EPI Signal	SDRAM Signal[1]	
EPI0S0	A0	D0
EPI0S1	A1	D1
EPI0S2	A2	D2
EPI0S3	A3	D3
EPI0S4	A4	D4
EPI0S5	A5	D5
EPI0S6	A6	D6
EPI0S7	A7	D7
EPI0S8	A8	D8
EPI0S9	A9	D9
EPI0S10	A10	D10
EPI0S11	A11	D11
EPI0S12	A12[2]	D12
EPI0S13	BA0	D13
EPI0S14	BA1	D14
EPI0S15	D15	
EPI0S16	DQML	
EPI0S17	DQMH	
EPI0S18	$\overline{CAS}$	
EPI0S19	$\overline{RAS}$	
EPI0S20-EPI0S27	not used	
EPI0S28	$\overline{WE}$	
EPI0S29	$\overline{CS}$	
EPI0S30	CKE	
EPI0S31	CLK	

(1) If 2 signals are listed, connect the EPI signal to both pins.
(2) Only for 256/512 megabit SDRAMs.

17.6.2 Refresh Configuration

The refresh count is based on the external clock speed and the number of rows per bank as well as the refresh period. The RFSH field represents how many external clock cycles remain before an AUTO-REFRESH is required. The normal formula is:

RFSH = ($t_{Refresh\_\mu s}$ / number_rows) /ext_clock_period

A refresh period is normally 64 ms, or 64000 µs. The number of rows is normally 4096 or 8192.The ext_clock_period is a value expressed in µsand is derived by dividing 1000 by the clock speed expressed in MHz. So, 50 MHz is 1000/50=20 ns, or 0.02 µs. A typical SDRAM is 4096 rows per bank if the system clock is running at 50 MHz with an EPIBAUD register value of 0:

RFSH = (64000/4096) / 0.02 = 15.625 µs /0.02 µs = 781.25

Copyright © 2012–2019, Texas Instruments Incorporated

The default value in the RFSH field is 750 decimal or 0x2EE to allow for a margin of safety and providing 15 µs per refresh. It is important to note that this number should always be smaller or equal to what is required by the above equation. For example, if running the external clock at 25 MHz (40 ns per clock period), 390 is the highest number that may be used. Note that the external clock may be 25 MHz when the system clock is 25 MHz or when the system clock is 50 MHz and configuring the COUNT0 field in the EPIBAUD register to 1 (divide by 2).

If a number larger than allowed is used, the SDRAM is not refreshed often enough, and data is lost.

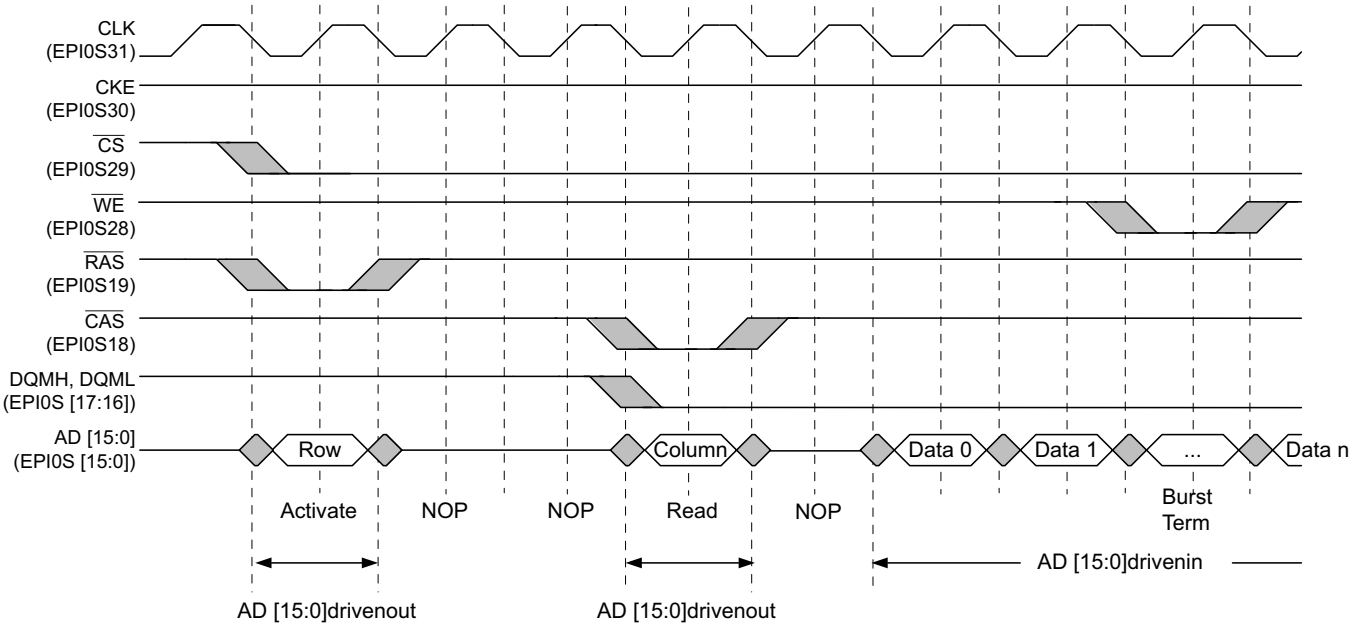
17.6.3 Bus Interface Speed

The EPI Controller SDRAM interface can operate up to 62.5 MHz. The COUNT0 field in the EPIBAUD register configures the speed of the EPI clock. For system clock (SYSCLK) speeds up to 62.5 MHz, the COUNT0 field can be 0x0000, and the SDRAM interface can run at the same speed as SYSCLK. However, if SYSCLK is running at higher speeds, the bus interface can run only as fast as half speed, and the COUNT0 field must be configured to at least 0x0001.

17.6.4 Non-Blocking Read Cycle

Figure 17-2 shows a non-blocking read cycle of n halfwords; n can be any number greater than or equal to 1. The cycle begins with the Activate command and the row address on the EPI0S[15:0] signals. With the programmed $\overline{CAS}$ latency of 2, the Read command with the column address on the EPI0S[15:0] signals follows after 2 clock cycles. Following one more NOP cycle, data is read in on the EPI0S[15:0] signals on every rising clock edge. The Burst Terminate command is issued during the cycle when the next-to-last halfword is read in. The DQMH and DQML signals are deasserted after the last halfword of data is received; the $\overline{CS}$ signal deasserts on the following clock cycle, signaling the end of the read cycle. At least one clock period of inactivity separates any two SDRAM cycles..

Figure 17-2. SDRAM Non-Blocking Read Cycle

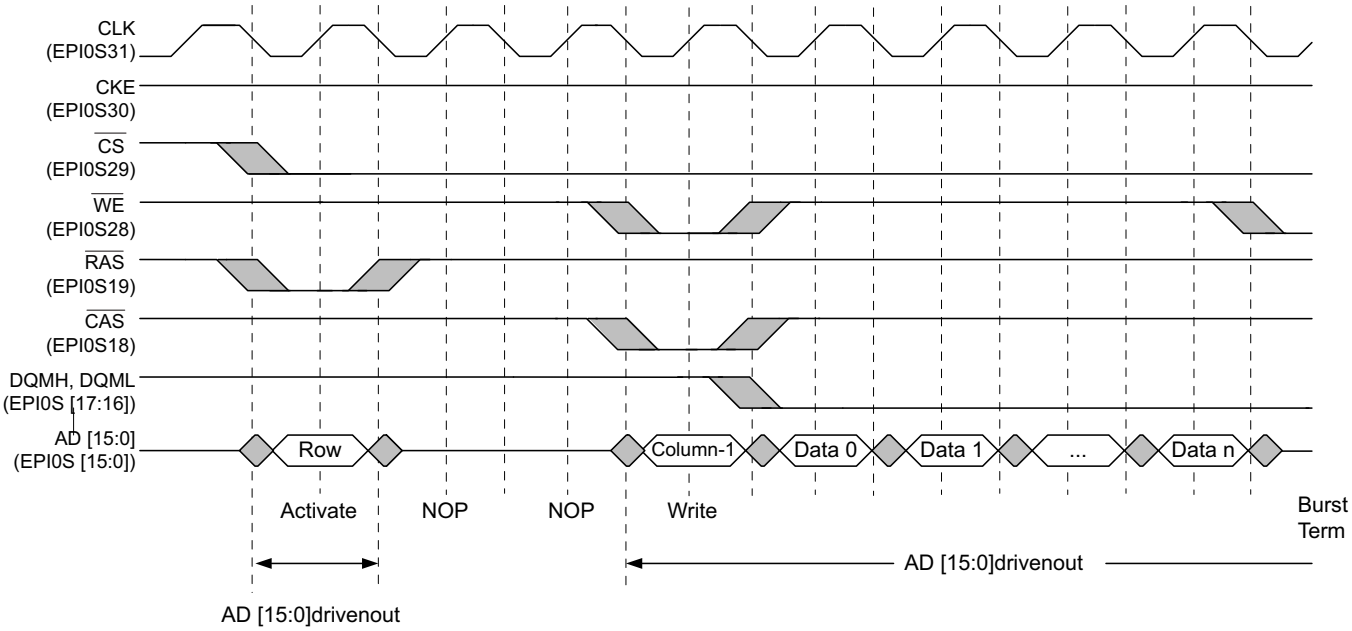


17.6.5 Normal Read Cycle

Figure 17-3 shows a normal read cycle of n halfwords; n can be 1 or 2. The cycle begins with the Activate command and the row address on the EPI0S[15:0] signals. With the programmed $\overline{CAS}$ latency of 2, the Read command with the column address on the EPI0S[15:0] signals follows after two clock cycles. Following one more NOP cycle, data is read in on the EPI0S[15:0] signals on every rising clock edge. The DQMH, DQML, and $\overline{CS}$ signals are deasserted after the last halfword of data is received, signaling the end of the cycle. At least one clock period of inactivity separates any two SDRAM cycles.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 17-3. SDRAM Normal Read Cycle

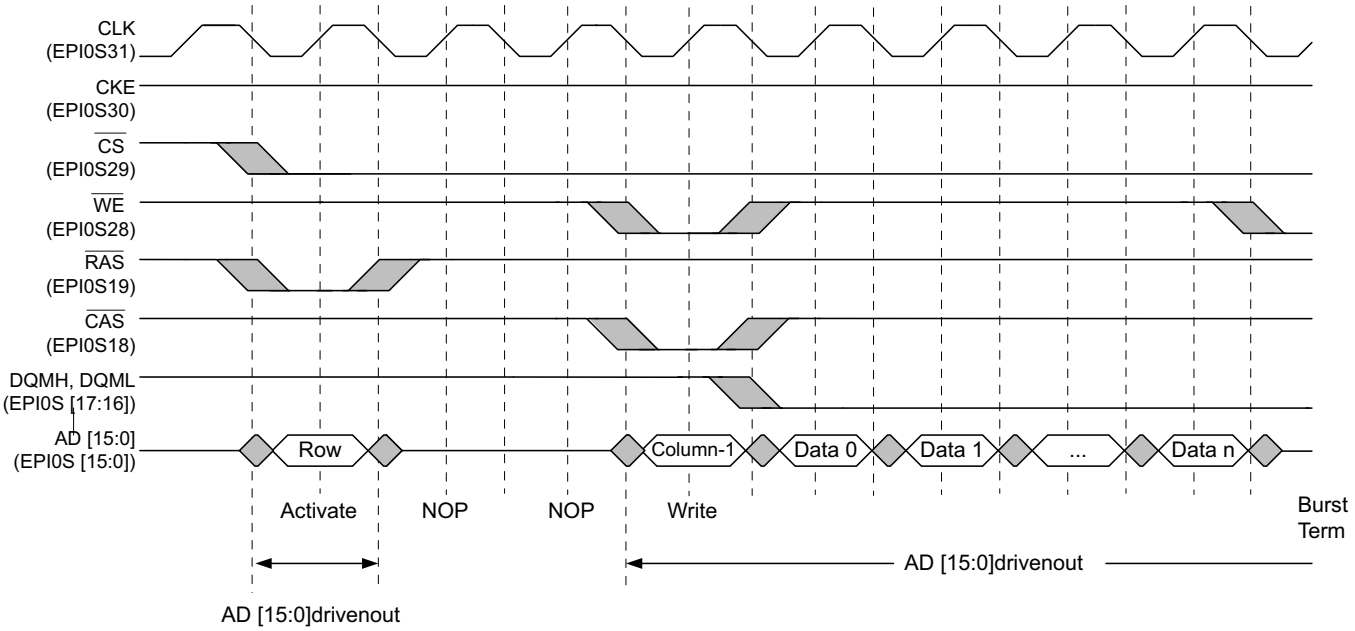


17.6.6 Write Cycle

Figure 17-4 shows a write cycle of n halfwords; n can be any number greater than or equal to 1. The cycle begins with the Activate command and the row address on the EPI0S[15:0] signals. With the programmed $\overline{\text{CAS}}$ latency of 2, the Write command with the column address on the EPI0S[15:0] signals follows after two clock cycles. When writing to SDRAMs, the Write command is presented with the first halfword of data. Because the address lines and the data lines are multiplexed, the column address is modified to be (programmed address -1). During the Write command, the DQMH and DQML signals are high, so no data is written to the SDRAM. On the next clock, the DQMH and DQML signals are asserted, and the data associated with the programmed address is written. The Burst Terminate command occurs during the clock cycle following the write of the last halfword of data. The $\overline{\text{WE}}$, DQMH, DQML, and $\overline{\text{CS}}$ signals are deasserted after the last halfword of data is received, signaling the end of the access. At least one clock period of inactivity separates any two SDRAM cycles.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 17-4. SDRAM Write Cycle



17.7 Host Bus Mode

The host bus supports the traditional 8-bit and 16-bit interfaces popularized by the 8051 devices and SRAM devices. This interface is asynchronous and uses strobe pins to control activity. Addressable memory can be doubled using Host Bus-16 mode as it performs half-word accesses. EPI0S0 should be connected to A0 of 16-bit memories.

17.7.1 Control Pins

The main three strobes are Address Latch Enable (ALE), Write ($\overline{WR}$), and Read ($\overline{RD}$, sometimes called $\overline{OE}$). Note that the timings are designed for older logic and so are hold-time vs. setup-time specific. The polarity of the read and write strobes can be active High or active Low by clearing or setting the RDHIGH and WRHIGH bits in the EPI Host-Bus n Configuration (EPIHBnCFGn) register.

The ALE can be changed to an active-low chip select signal, $\overline{CS}$, through the EPIHBnCFGn register. The ALE is best used for Host-Bus muxed mode in which EPI address and data pins are shared. All Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an external latch to capture the address then hold it until the data phase. The polarity of the ALE can be active High or Low by clearing or setting the ALEHIGH bit in the EPI Host-Bus n Configuration (EPIHBnCFGn) register. $\overline{CS}$ is best used for Host-Bus unmuxed mode in which EPI address and data pins are separate. The $\overline{CS}$ indicates when the address and data phases of a read or write access are occurring. Both the ALE and the $\overline{CS}$ modes can be enhanced to access four external devices using settings in the EPIHBnCFGn register. Wait states can be added to the data phase of the access using the WRWS and RDWS bits in the EPIHBnCFGn register. Additionally, within these wait state options, the WRWSM and RDWSM bit of the EPIHBnTIMEn register can be set to reduce the given wait states by 1 EPI clock cycle for finer granularity.

For FIFO mode, the ALE is not used, and two input holds are optionally supported to gate input and output to what the XFIFO can handle.

Copyright © 2012–2019, Texas Instruments Incorporated

Host-Bus 8 and Host-Bus 16 modes are very configurable. The user has the ability to connect 1,2, or 4 external devices to the EPI signals, as well as control whether byte select signals are provided in HB16 mode. These capabilities depend on the configuration of the MODE field in the EPIHBnCFG register, the CSCFG field and the CSCFGEXT bit in the EPIHBnCFGn register, and the BSEL bit in the EPIHB16CFG register. The CSCFGEXT bit extends the chip select configuration possibilities by providing the most significant bit of the CSCFG field. Refer to Table 17-2 for the possible ALE and chip select options that can be programmed by the combination of the CSCFGEXT and CSCFG bits. Note that CSCFGEXT is the most significant bit.

Table 17-2. CSCFGEXT + CSCFG Encodings

Value	Description
0x0	ALE Configuration EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (MODE field in the EPIHB8CFG register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle.
0x1	$\overline{CS}$ Configuration EPI0S30 is used as a Chip Select ($\overline{CS}$). When using this mode, the address and data are generally not muxed (MODE field in the EPIHB8CFG register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when $\overline{CS}$ is low.
0x2	Dual $\overline{CS}$ Configuration EPI0S30 is used as $\overline{CS0}$ and EPI0S27 is used as $\overline{CS1}$. Whether $\overline{CS0}$ or $\overline{CS1}$ is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral.
0x3	ALE with Dual $\overline{CS}$ Configuration EPI0S30 is used as address latch (ALE), EPI0S27 is used as $\overline{CS1}$, and EPI0S26 is used as $\overline{CS0}$. Whether $\overline{CS0}$ or $\overline{CS1}$ is asserted is determined by the most significant address bit for a respective external address map.
0x4	ALE with Single $\overline{CS}$ Configuration EPI0S30 is used as address latch (ALE) and EPI0S27 is used as $\overline{CS}$
0x5	Quad $\overline{CS}$ Configuration EPI0S30 is used as $\overline{CS0}$ and EPI0S27 is used as $\overline{CS1}$. EPI0S34 is used as $\overline{CS2}$ and EPI0S33 is used as $\overline{CS3}$
0x6	ALE with Quad $\overline{CS}$ Configuration EPI0S30 is used as $\overline{CS0}$ and EPI0S27 is used as $\overline{CS1}$. EPI0S34 is used as $\overline{CS2}$ and EPI0S33 is used as $\overline{CS3}$
0x7	Tri $\overline{CS}$ Configuration. EPI0S30 is used $\overline{CS0}$. EPI0S34 is used as $\overline{CS2}$ and EPI0S33 is used as $\overline{CS3}$

If one of the dual-chip-select modes is selected (CSCFGEXT is 0x0 and CSCFG is 0x2 or 0x3 in the EPIHBnCFGn register), both chip selects can share the peripheral, code, or the memory space, or one chip select can use the peripheral space and the other can use the memory or code space.

In the EPIADDRMAP register, if the EPADR field is not 0x0, the ECADR field is 0x0, and the ERADR field is 0x0, then the address specified by EPADR is used for both chip selects, with $\overline{CS0}$ being asserted when the MSB of the address range is 0 and $\overline{CS1}$ being asserted when the MSB of the address range is 1.

If the ERADR field is not 0x0, the ECADR field is 0x0, and the EPADR field is 0x0, then the address specified by ERADR is used for both chip selects, with the MSB performing the same delineation.

If both the EPADR and the ERADR are not 0x0, and the ECADR field is 0x0 and the EPI is configured for dual-chip selects, then $\overline{CS0}$ is asserted for either address range defined by EPADR and $\overline{CS1}$ is asserted for either address range defined by ERADR.

The two chip selects can also be shared between the code space and memory or peripheral space. If the ECADR field is 0x1, ERADR field is 0x0, and the EPADR field is not 0x0, then $\overline{CS0}$ is asserted for the address range defined by ECADR and $\overline{CS1}$ is asserted for either address range defined by EPADR.

If the ECADR field is 0x1, EPADR field is 0x0, and the ERADR field is not 0x0, then $\overline{CS0}$ is asserted for the address range defined by ECADR and $\overline{CS1}$ is asserted for either address range defined by ERADR.

Copyright © 2012–2019, Texas Instruments Incorporated

In quad chip select mode (CSCFGEXT is 0x1 and CSCFG is 0x1 or 0x2 in the EPIHBnCFG2 register), both the peripheral and the memory space must be enabled.

In the EPIADDRMAP register, the EPADR field is 0x3, the ERADR field is 0x3, and the ECADR field is 0x0. With this configuration, $\overline{CS0}$ asserts for the address range beginning at 0x6000.0000, $\overline{CS1}$ asserts for 0x8000.0000, $\overline{CS2}$ for 0xA000.0000, and $\overline{CS3}$ for 0xC000.0000.

Table 17-3 gives a detailed explanation of chip select address range mappings based on combinations of enabled peripheral and memory space.

Table 17-3. Dual- and Quad- Chip Select Address Mappings

Chip Select Mode	ERADR	EPADR	ECADR	$\overline{CS0}$[1]	$\overline{CS1}$	$\overline{CS2}$	$\overline{CS3}$
Dual-chip select	0x0	0x1 or 0x2	0x0	EPADR defined address range (0xA000.0000 or 0xC000.0000)	EPADR defined address range (0xA000.0000 or 0xC000.0000)	N/A	N/A
Dual-chip select	0x1 or 0x2	0x0	0x0	ERADR defined address range (0x6000.0000 or 0x8000.000)	ERADR defined address range (0x6000.0000 or 0x8000.000)	N/A	N/A
Dual-chip select	0x1 or 0x2	0x1 or 0x2	0x0	EPADR defined address range (0xA000.0000 or 0xC000.0000)	ERADR defined address range (0x6000.0000 or 0x8000.0000)	N/A	N/A
Dual-chip select	0x0	0x1 or 0x2	0x1	ECADR defined address range (0x1000.0000)	EPADR defined address range (0xA000.0000 or 0xC000.0000)	N/A	N/A
Dual-chip select	0x1 or 0x2	0x0	0x1	ECADR defined address range (0x1000.0000)	ERADR defined address range (0x6000.0000 or 0x8000.0000)	N/A	N/A
Quad-chip select	0x3	0x3	0x0	0x6000.0000	0x8000.0000	0xA000.0000	0xC000.0000
Tri-chip select	0x3	0x3	0x0	0x6000.0000	N/A	0xA000.0000	0xC000.0000

(1) When $\overline{CS0}$ and CS1 share address space, $\overline{CS0}$ asserts when the MSB of the address is '0' and $\overline{CS1}$, when the MSB of the address is '1'.

The MODE field of the EPIHBnCFGn registers configure the interface for the chip selects, which support ADMUX or ADNOMUX. See Table 17-4 for details on which configuration register controls each chip select. If the CSBAUD bit is clear, all chip selects are configured by the MODE bit field of the EPIHBnCFG register.

If the CSBAUD bit in the EPIHBnCFG2 register is set in dual-chip select mode, the two chip selects can use different clock frequencies, wait states and strobe polarity. If the CSBAUD bit is clear, both chip selects use the clock frequency, wait states, and strobe polarity defined for $\overline{CS0}$. Additionally, if the CSBAUD bit is set, the two chip selects can use different interface modes. If any interface modes are programmed to ADMUX, then dual chip select mode must include the ALE capability. In quad chip select mode, if the CSBAUD bit in the EPIHBnCFG2 register is set, the four chip selects can use different clock frequencies, wait states and strobe polarity. If the CSBAUD bit is clear, all chip selects use the clock frequency, wait states, and strobe polarity defined for $\overline{CS0}$. If the CSBAUD bit is set, the four chip selects can use different interface modes

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-4. Chip Select Configuration Register Assignment

Configuration Register[1]	Corresponding Chip Select
EPIHBnCFG	$\overline{CS0}$
EPIHBnCFG2	$\overline{CS1}$
EPIHBnCFG3	$\overline{CS2}$
EPIHBnCFG4	$\overline{CS3}$

(1) If the CSBAUD bit in the EPIHBnCFG2 register is clear and multiple chip selects are enabled, then all chip selects are configured by the MODE bit field in the EPIHBnCFG register.

Note that multiple chip select modes do not allow the intermixing of Host-Bus 8 and Host-Bus16 modes.

When BSEL=1 in the EPIHB16CFG register, byte select signals are provided, so byte-sized data can be read and written at any address; however, these signals reduce the available address width by two pins. The byte select signals are active Low. $\overline{BSEL0}$ corresponds to the LSB of the halfword, and $\overline{BSEL1}$ corresponds to the MSB of the halfword.

When BSEL=0, byte reads and writes at odd addresses only act on the even byte, and byte writes at even addresses write invalid values into the odd byte. As a result, accesses should be made as half-words (16-bits) or words (32-bits). In C/C++, programmers should use only short int and long int for accesses. Also, because data accesses in HB16 mode with no byte selects are on 2-byte boundaries, the available address space is doubled. For example, 28 bits of address accesses 512 MB in this mode. Table 17-5 shows the capabilities of the HB8 and HB16 modes as well as the available address bits with the possible combinations of these bits.

Although the EPI0S31 signal can be configured for the EPI clock signal in Host-Bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system.

Table 17-5. Capabilities of Host Bus 8 and Host Bus 16 Modes

Host Bus Type	Mode	CSCFGEXT	CSCFG	Max # of External Devices	BSEL	Byte Access	Available Address	Addressable Memory
HB8	0x0	0	0x0,0x1	1	N/A	Always	28	256 MB
HB8	0x0	0	0x2	2	N/A	Always	27	128 MB
HB8	0x0	0	0x3	2	N/A	Always	26	64 MB
HB8	0x0	1	0x0	1	N/A	Always	27	128 MB
HB8	0x0	1	0x1	4	N/A	Always	27	128 MB
HB8	0x0	1	0x2	4	N/A	Always	26	64 MB
HB8	0x1	0	0x0,0x1	1	N/A	Always	20	1 MB
HB8	0x1	0	0x2	2	N/A	Always	19	512 KB
HB8	0x1	0	0x3	2	N/A	Always	18	256 KB
HB8	0x1	1	0x0	1	N/A	Always	19	512 MB
HB8	0x1	1	0x1	4	N/A	Always	19	512 MB
HB8	0x1	1	0x2	4	N/A	Always	18	256 MB
HB8	0x3	X	X	-	N/A	Always	None	-
HB16	0x0	0	0x0,0x1	1	0	No	28 (1)	512 MB
HB16	0x0	0	0x0,0x1	1	1	Yes	26(2)	64 MB
HB16	0x0	0	0x2	2	0	No	27(1)	256 MB
HB16	0x0	0	0x2	2	1	Yes	25(2)	32 MB
HB16	0x0	0	0x3	2	0	No	26(1)	128 MB
HB16	0x0	0	0x3	2	1	Yes	24(2)	16 MB
HB16	0x0	1	0x0	1	0	No	27(1)	256 MB
HB16	0x0	1	0x0	1	1	Yes	25(2)	32 MB
HB16	0x0	1	0x1	4	0	No	27(1)	256 MB

(1) If byte selects are not used, data accesses are on 2-byte boundaries. As a result, the available address space is doubled.
(2) Two EPI signals are used for byte selects, reducing the available address space by two bits.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-5. Capabilities of Host Bus 8 and Host Bus 16 Modes (continued)

Host Bus Type	Mode	CSCFGEXT	CSCFG	Max # of External Devices	BSEL	Byte Access	Available Address	Addressable Memory
HB16	0x0	1	0x1	4	1	Yes	25[2]	32 MB
HB16	0x0	1	0x2	4	0	No	26[1]	128 MB
HB16	0x0	1	0x2	4	1	Yes	24[2]	16 MB
HB16	0x1	0	0x0,0x1	1	0	No	12[1]	8 KB
HB16	0x1	0	0x0,0x1	1	1	Yes	10[2]	1 KB
HB16	0x1	0	0x2	2	0	No	11[1]	4 KB
HB16	0x1	0	0x2	2	1	Yes	9[2]	512 B
HB16	0x1	0	0x3	2	0	No	10[1]	2 KB
HB16	0x1	0	0x3	2	1	Yes	8[2]	256 B
HB16	0x1	1	0x0	1	0	No	11[1]	4 KB
HB16	0x1	1	0x0	1	1	Yes	9[2]	256 B
HB16	0x1	1	0x1	4	0	No	17[1]	128 MB
HB16	0x1	1	0x1	4	1	Yes	15[2]	32 KB
HB16	0x1	1	0x2	4	0	No	10[1]	2 KB
HB16	0x1	1	0x2	4	1	Yes	8[2]	256 B
HB16	0x1	1	0x3	3	0	No	19[1]	1 MB
HB16	0x1	1	0x3	3	1	Yes	17[2]	128 KB
HB16	0x3	X	X	-	X	-	None	-

Table 17-6 shows how the EPI[35:0] signals function while in Host-Bus 8 mode. Notice that the signal configuration changes based on the address/data mode selected by the MODE field in the EPIHB8CFGn register and on the chip select configuration selected by the CSCFG and CSCFGEXT field in the EPIHB8CFG2 register.

Although the EPI0S31 signal can be configured for the EPI clock signal in Host-Bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-6. EPI Host-Bus 8 Signal Connections[1][2]

EPI Signal	CSCFG	HB8 Signal (MODE= ADMUX)	HB8 Signal (MODE = ADNOMUX (Cont.Read))	HB8 Signal (MODE= XFIFO)
EPI0S0	X	AD0	D0	D0
EPI0S1	X	AD1	D1	D1
EPI0S2	X	AD2	D2	D2
EPI0S3	X	AD3	D3	D3
EPI0S4	X	AD4	D4	D4
EPI0S5	X	AD5	D5	D5
EPI0S6	X	AD6	D6	D6
EPI0S7	X	AD7	D7	D7
EPI0S8	X	A8	A0	-
EPI0S9	X	A9	A1	-
EPI0S10	X	A10	A2	-
EPI0S11	X	A11	A3	-
EPI0S12	X	A12	A4	-
EPI0S13	X	A13	A5	-
EPI0S14	X	A14	A6	-
EPI0S15	X	A15	A7	-
EPI0S16	X	A16	A8	-
EPI0S17	X	A17	A9	-
EPI0S18	X	A18	A10	-
EPI0S19	X	A19	A11	-
EPI0S20	X	A20	A12	-
EPI0S21	X	A21	A13	-
EPI0S22	X	A22	A14	-
EPI0S23	X	A23	A15	-
EPI0S24	X	A24	A16	-
EPI0S25	0x0	A25	A17	-
	0x1			
	0x2			$\overline{CS1}$
	0x3			-
	0x4			-
	0x5			-
	0x6			-
EPI0S26	0x0	A26	A18	FEMPTY
	0x1			
	0x2			
	0x3	$\overline{CS0}$	$\overline{CS0}$	
	0x4	A26	A18	
	0x5			
	0x6	$\overline{CS0}$	$\overline{CS0}$	

(1) "X" indicates the state of this field is a don't care.
(2) When an entry straddles several rows, the signal configuration is the same for all rows.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-6. EPI Host-Bus 8 Signal Connections[1][2] (continued)

EPI Signal	CSCFG	HB8 Signal (MODE= ADMUX)	HB8 Signal (MODE = ADNOMUX (Cont.Read))	HB8 Signal (MODE= XFIFO)
EPI0S27	0x0	A27	A19	FFULL
	0x1			
	0x2	$\overline{CS1}$	$\overline{CS1}$	
	0x3			
	0x4	$\overline{CS0}$	$\overline{CS0}$	
	0x5	$\overline{CS1}$	$\overline{CS1}$	
	0x6			
EPI0S28	X	$\overline{RD}/\overline{OE}$	$\overline{RD}/\overline{OE}$	$\overline{RD}$
EPI0S29	X	$\overline{WR}$	$\overline{WR}$	$\overline{WR}$
EPI0S30	0x0	ALE	ALE	-
	0x1	$\overline{CS}$	$\overline{CS}$	$\overline{CS}$
	0x2	$\overline{CS0}$	$\overline{CS0}$	$\overline{CS0}$
	0x3	ALE	ALE	-
	0x4			-
	0x5	$\overline{CS0}$	$\overline{CS0}$	-
	0x6	ALE	ALE	-
EPI0S31	X	Clock[3]	Clock[3]	Clock[3]
EPIOS32	X	iRDY	iRDY	iRDY
EPIOS33	0x0	X	X	X
	0x1	X	X	X
	0x2	X	X	X
	0x3	X	X	X
	0x4	X	X	X
	0x5	$\overline{CS3}$	$\overline{CS3}$	X
	0x6			X
EPI0S34	0x0	X	X	X
	0x1	X	X	X
	0x2	X	X	X
	0x3	X	X	X
	0x4	X	X	X
	0x5	$\overline{CS2}$	$\overline{CS2}$	X
	0x6			X
EPI0S35	0x0	X	X	X
	0x1	X	X	X
	0x2	X	X	X
	0x3	X	X	X
	0x4	X	X	X
	0x5	CRE	CRE	X
	0x6			X

(3) The clock signal is not required for this mode and has unspecified timing relationships to other signals.

Table 17-7 shows how the EPI[41:0] signals function while in Host-Bus 16 mode. Notice that the signal configuration changes based on the address/data mode selected by the MODE field in the EPIHB16CFG2 register, on the chip select configuration selected by the CSCFG field in the same register, and on whether byte selects are used as configured by the BSEL bit in the EPIHB16CFG register. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-7. EPI Host-Bus 16 Signal Connections[1][2]

EPI Signal	{CSCFGEXT,CSCFG}	BSEL	HB16 Signal (MODE=ADMUX)[3]	HB16 Signal (MODE-ADNOMUX)	B16 Signal MODE=XFIFO)
EPI0S0	X	X	AD0	D0	D0
EPI0S1	X	X	AD1	D1	D1
EPI0S2	X	X	AD2	D2	D2
EPI0S3	X	X	AD3	D3	D3
EPI0S4	X	X	AD4	D4	D4
EPI0S5	X	X	AD5	D5	D5
EPI0S6	X	X	AD6	D6	D6
EPI0S7	X	X	AD7	D7	D7
EPI0S8	X	X	AD8	D8	D8
EPI0S9	X	X	AD9	D9	D9
EPI0S10	X	X	AD10	D10	D10
EPI0S11	X	X	AD11	D11	D11
EPI0S12	X	X	AD12	D12	D12
EPI0S13	X	X	AD13	D13	D13
EPI0S14	X	X	AD14	D14	D14
EPI0S15	X	X	AD15	D15	D15
EPI0S16	X	X	A16	A0[3]	-
EPI0S17	X	X	A17	A1	-
EPI0S18	X	X	A18	A2	-
EPI0S19	X	X	A19	A3	-
EPI0S20	X	X	A20	A4	-
EPI0S21	X	X	A21	A5	-
EPI0S22	X	X	A22	A6	-
EPI0S23	X	X	A23	A7	-
EPI0S24	0x0,0x1,0x2	0	A24	A8	-
		1			
	0x3	0			
		1	$\overline{BSEL0}$	$\overline{BSEL0}$	
	0x4,0x5	0	A24	A8	
		1			
	0x6	0			
		1	$\overline{BSEL0}$	$\overline{BSEL0}$	
	0x7	0	-	A8	
		1			
EPI0S25	0x0, 0x1	X	A25	A9	-
	0x2	0	A25	A9	$\overline{CS1}$
		1	$\overline{BSEL0}$	$\overline{BSEL0}$	
	0x3	0	A25	A9	
		1	$\overline{BSEL1}$	$\overline{BSEL1}$	
	0x4	0	A25	A9	
		1	$\overline{BSEL0}$	$\overline{BSEL0}$	
	0x5	0	A25	A9	
		1	$\overline{BSEL0}$	$\overline{BSEL0}$	

(1) "X" indicates the state of this field is a don't care.
(2) When an entry straddles several row, the signal configuration is the same for all rows.
(3) In this mode, halfword accesses are used. A0 is the LSB of the address and is equivalent to the internal Cortex-M3 A1 address. This pin should be connected to A0 of 16-bit memories.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-7. EPI Host-Bus 16 Signal Connections[1][2] (continued)

EPI Signal	{CSCFGEXT,CSCFG}	BSEL	HB16 Signal (MODE=ADMUX)[3]	HB16 Signal (MODE=ADNOMUX)	B16 Signal MODE=XFIFO)
	0x6	0	A25	A9	
		1	$\overline{\text{BSEL1}}$	$\overline{\text{BSEL1}}$	
	0x7	0	-	A9	
		1		$\overline{\text{BSEL0}}$	
EPI0S26	0x0	0	A26	A10	FEMPTY
		1	$\overline{\text{BSEL0}}$	$\overline{\text{BSEL0}}$	
	0x1	0	A26	A10	
		1	$\overline{\text{BSEL0}}$	$\overline{\text{BSEL0}}$	
	0x2	0	A26	A10	
		1	$\overline{\text{BSEL1}}$	$\overline{\text{BSEL1}}$	
	0x3	X	$\overline{\text{CS0}}$	$\overline{\text{CS0}}$	
	0x4	0	A26	A10	-
		1	$\overline{\text{BSEL1}}$	$\overline{\text{BSEL1}}$	
	0x5	0	A26	A10	
		1	$\overline{\text{BSEL1}}$	$\overline{\text{BSEL1}}$	
	0x6	X	$\overline{\text{CS0}}$	$\overline{\text{CS0}}$	
	0x7	0	-	A10	
		1		$\overline{\text{BSEL1}}$	
EPI0S27	0x0	0	A27	A11	FFULL
		1	$\overline{\text{BSEL1}}$	$\overline{\text{BSEL1}}$	
	0x1	0	A27	A11	
		1	$\overline{\text{BSEL1}}$	$\overline{\text{BSEL1}}$	
	0x2	X	$\overline{\text{CS1}}$	$\overline{\text{CS1}}$	
	0x3	X	$\overline{\text{CS1}}$	$\overline{\text{CS1}}$	
	0x4	X	$\overline{\text{CS0}}$	$\overline{\text{CS0}}$	-
	0x5	X	$\overline{\text{CS1}}$	$\overline{\text{CS1}}$	
	0x6	X	$\overline{\text{CS1}}$	$\overline{\text{CS1}}$	
	0x7	0	-	A20	
		1		A18	
EPI0S28	X	X	$\overline{\text{RD}}/\overline{\text{OE}}$	$\overline{\text{RD}}/\overline{\text{OE}}$	$\overline{\text{RD}}$
EPI0S29	X	X	$\overline{\text{WR}}$	$\overline{\text{WR}}$	$\overline{\text{WR}}$
EPI0S30	0x0	X	ALE	ALE	-
	0x1	X	$\overline{\text{CS}}$	$\overline{\text{CS}}$	$\overline{\text{CS}}$
	0x2	X	$\overline{\text{CS0}}$	$\overline{\text{CS0}}$	$\overline{\text{CS0}}$
	0x3	X	ALE	ALE	-
	0x4	X	ALE	ALE	-
	0x5	X	$\overline{\text{CS0}}$	$\overline{\text{CS0}}$	-
	0x6	X	ALE	ALE	-
	0x7	X	$\overline{\text{CS0}}$	$\overline{\text{CS0}}$	-
EPI0S31	X	X	CLK	CLK	CLK
EPI0S32	X	X	iRDY	iRDY	iRDY
EPI0S33	X	X	$\overline{\text{CS3}}$	$\overline{\text{CS3}}$	-
EPI0S34	X	X	$\overline{\text{CS2}}$	$\overline{\text{CS2}}$	-
EPI0S35	0x0-0x6	X	CRE	CRE	-
	0x7	0	-	A19	
		1		A17	

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-7. EPI Host-Bus 16 Signal Connections[1][2] (continued)

EPI Signal	{CSCFGEXT,CSCFG}	BSEL	HB16 Signal (MODE=ADMUX)[3]	HB16 Signal (MODE=ADNOMUX)	B16 Signal MODE=XFIFO)
EPI0S36	0x0-0x4	X	-	-	-
	0x5	X	-	A11	
	0x6	X	-	-	
	0x7	X	-	A11	
EPI0S37	0x0-0x4	X	-	-	
	0x5	X	-	A12	
	0x6	X	-	-	
	0x7	X	-	A12	
EPI0S38	0x0-0x4	X	-	-	
	0x5	X	-	A13	
	0x6	X	-	-	
	0x7	X	-	A13	
EPI0S39	0x0-0x4	X	-	-	
	0x5	X	-	A14	
	0x6	X	-	-	
	0x7	X	-	A14	
EPI0S40	0x0-0x4	X	-	-	
	0x5	X	-	A15	
	0x6	X	-	-	
	0x7	X	-	A15	
EPI0S41	0x0-0x4	X	-	-	
	0x5	X	-	A16	
	0x6	X	-	-	
	0x7	X	-	A16	

The RDYEN in the EPIHBnCFG enables the monitoring of the external iRDY pin to stall synchronous accesses. On the rising edge of EPI clock, if iRDY is low, access is stalled. The IRDYDLY can program the number of EPI clock cycles in advance to the stall (1,2, or 3) as shown in Figure 17-5. This is a conceptual timing diagram of how the iRDY signal works with different IRDYDLY configurations. When enabled, the iRDY stalls the EPI's internal states, while IRDYDLY controls the delay pipeline when this stall takes affect. The iRDY signal can be connected to multiple devices with a pull up resistor as shown in Figure 17-6.

Figure 17-7 shows how to connect the EPI signals to a16-bit SRAM and a 16-bit Flash memory with muxed address and memory using byte selects and dual chip selects with ALE. This schematic is just an example of how to connect the signals; timing and loading have not been analyzed. In addition, not all bypass capacitors are shown.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 17-5. iRDY Access Stalls

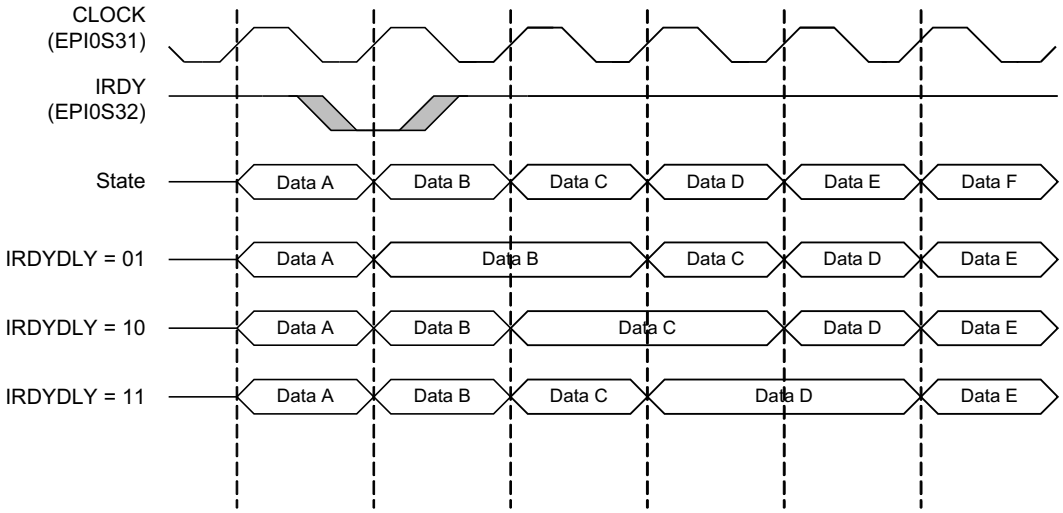
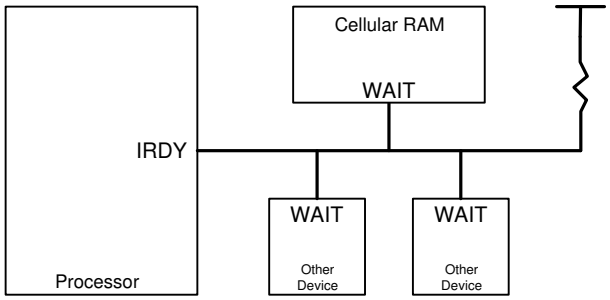
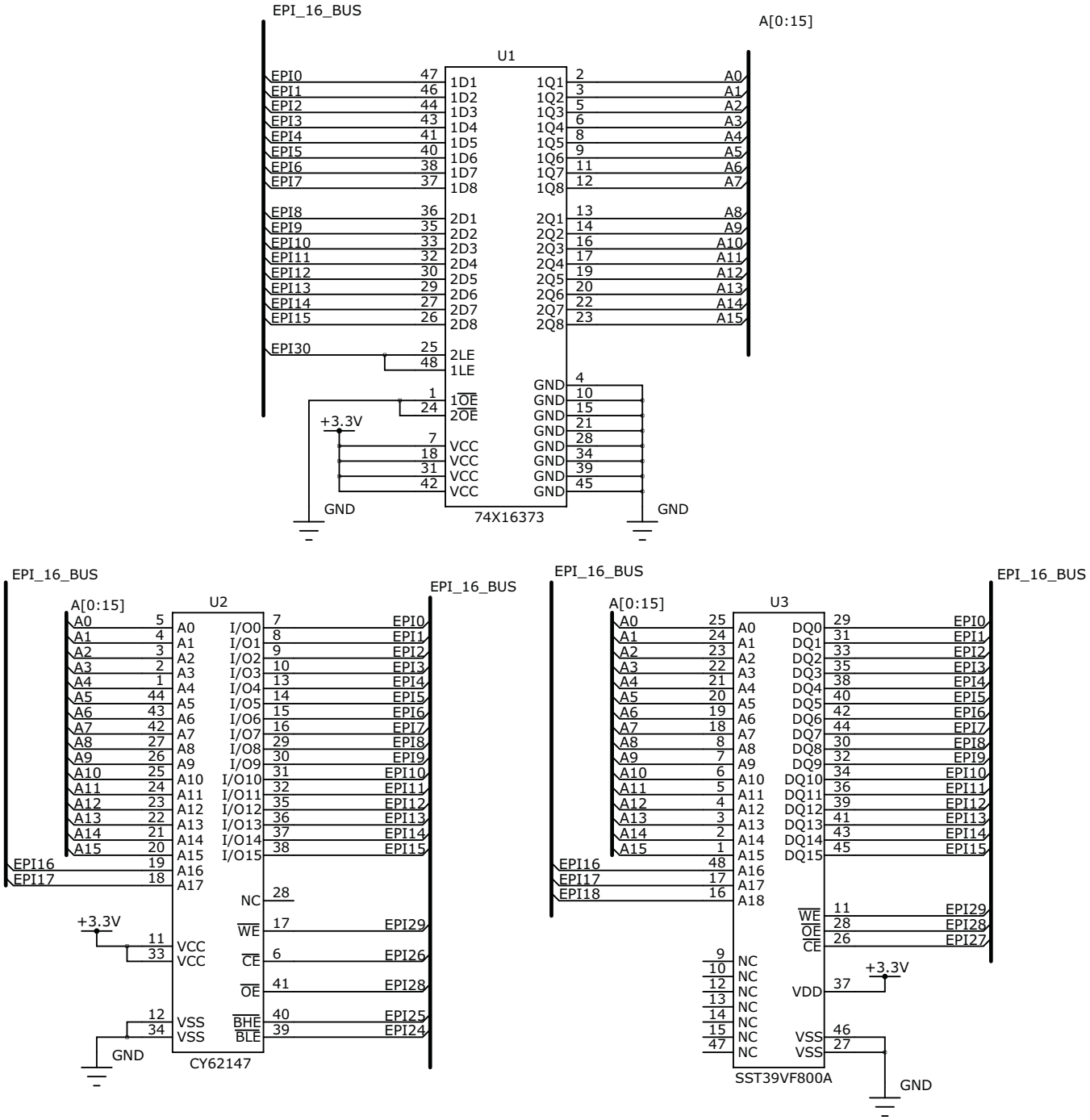


Figure 17-6. iRDY Signal Connection



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 17-7. Example Schematic for Muxed Host-Bus 16 Mode



17.7.2 Speed of Transactions

The COUNT0 field in the EPIBAUD register must be configured to set the main transaction rate based on what the slave device can support (including wiring considerations). The main control transitions are normally ½ the baud rate (COUNT0 = 1) because the EPI block forces data vs. control to change on alternating clocks. When using dual chip selects, each chip select can access the bus using differing baud rates by setting the CSBAUD bit in the EPIHBnCFG2 register. In this case, the COUNT0 field controls the $\overline{CS0}$ transactions, and the COUNT1 field controls the $\overline{CS1}$ transactions. When using quad chip select mode, the COUNT0 bit field of the EPIBAUD2 register controls the baud rate of $\overline{CS2}$ and the COUNT1 bit field is programmed to control the baud rate of $\overline{CS3}$.

Copyright © 2012–2019, Texas Instruments Incorporated

Additionally, the Host-Bus mode provides read and write wait states for the data portion to support different classes of device. These wait states stretch the data period (hold the rising edge of data strobe) and may be used in all four sub-modes. The wait states are set using the WRWS and RDWS bits in the EPI Host-Bus n Configuration (EPIHBnCFGn) register. The WRWS and RDWS bits are enhanced with more precision by WRWSM and RDWSM bits in the EPIHBnTIMEn registers. Note none of the wait state configuration bits can be set concurrently with the BURST bit in the same EPIHBnCFGn register. See Table 17-8 for programming information.

Table 17-8. Data Phase Wait State Programming

RDWS or WRWS Encoding in EPIHBnCFGn Register	RDWSM or WRWSM Encoding in EPIHBnTIMEn Registers	Data Phase Wait States
0x0	1	1 EPI clocks
0x0	0	2EPI clocks
0x1	1	3 EPI clocks
0x1	0	4 EPI clocks
0x2	1	5 EPI clocks
0x2	0	6 EPI clocks
0x3	1	7 EPI clocks
0x3	0	8 EPI clocks

The CAPWIDTH bit in EPIHBnTIMEn registers controls the delay between Host-Bus transfers. When the CSBAUD bit is set and multi-chip selects have been configured in the EPIHBnCFG2 registers, delay takes an additional clock cycle to adjust the clock rate of different chip selects.

Word read and write transactions can be enhanced through the enabling of the BURST bit in the EPIHB16CFGn registers.

17.7.3 Sub-Modes of Host Bus 8/16

The EPI controller supports four variants of the Host-Bus model using 8 or 16 bits of data in all four cases. The four sub-modes are selected using the MODE bits in the EPIHBnCFG register, and are:

- Address and data are muxed. This scheme is used by many 8051 devices, some Microchip PIC parts, and some ATmega parts. When used for standard SRAMs, a latch must be used between the microcontroller and the SRAM. This sub-mode is provided for compatibility with existing devices that support data transfers without a latch (that is, CPLDs). In general, the de-muxed sub-mode should normally be used. The ALE configuration should be used in this mode, as all Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an external latch to capture the address then hold until the data phase. The ALE configuration is controlled by configuring the CSCFG and CSCFGEXT field to be 0x0 in the EPIHBnCFG2 register. The ALE can be enhanced to access two or four external devices with four separate $\overline{CS}$ signals. By configuring the CSCFG field to be 0x3 and the CSCFGEXT bit to be 0 in the EPIHBnCFG2 register, EPI0S30 functions as ALE, EPI0S27 functions as $\overline{CS1}$, and EPI0S26 functions as $\overline{CS0}$. When the CSCFG field is set to 0x0 and the CSCFGEXT bit is set to 1 in the EPIHBnCFG2 register, EPI0S30 functions as ALE, EPIOS33 functions as $\overline{CS3}$, EPIOS34 functions as $\overline{CS2}$, EPI0S27 functions as $\overline{CS1}$, and EPI0S26 functions as $\overline{CS0}$. The $\overline{CS}$ is best used for host-bus unmuxed mode, in which EPI address and data pins are separate. The $\overline{CS}$ indicates when the address and data phases of a read or write access are occurring.
- Address and data are separate with 8 or 16 bits of data and up to 20 bits of address (1 MB). This scheme is used by more modern 8051 devices, as well as some PIC and ATmega parts. This mode is generally used with real SRAMs, many EEPROMs, and many NOR Flash memory devices. Note that there is no hardware command write support for Flash memory devices; this mode should only be used for Flash memory devices programmed at manufacturing time. If a Flash memory device must be written and does not support a direct programming model, the command mechanism must be performed in software. The $\overline{CS}$ configuration should be used in this mode. The $\overline{CS}$ signal indicates when the address and data phases of a read or write access is occurring. The $\overline{CS}$ configuration is controlled by configuring the CSCFG field to be 0x1and the CSCFGEXT bit to be 0 in the EPIHBnCFG2 register.
- Continuous read mode where address and data are separate. This sub-mode is used for real SRAMs

Copyright © 2012–2019, Texas Instruments Incorporated

which can be read more quickly by only changing the address (and not using $\overline{RD}/\overline{OE}$ strobing). In this sub-mode, reads are performed by keeping the read mode selected (output enable is asserted) and then changing the address pins. The data pins are changed by the SRAM after the address pins change. For example, to read data from address 0x100 and then 0x101, the EPI controller asserts the output-enable signal and then configures the address pins to 0x100; the EPI controller then captures what is on the data pins and increments A0 to 1 (so the address is now 0x101); the EPI controller then captures what is on the data pins. Note that this mode consumes higher power because the SRAM must continuously drive the data pins. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available. Writes are not permitted in this mode.

- FIFO mode uses 8 or 16 bits of data, removes ALE and address pins and optionally adds external XFIFO FULL/EMPTY flag inputs. This scheme is used by many devices, such as radios, communication devices (including USB2 devices), and some FPGA configurations (FIFO through block RAM). This sub-mode provides the data side of the normal Host-Bus interface, but is paced by the FIFO control signals. It is important to consider that the XFIFO FULL/EMPTY control signals may stall the interface and could have an impact on blocking read latency from the processor or µDMA.

The WORD bit in the EPIHBnCFG2 register can be set to use memory more efficiently. By default, the EPI controller uses data bits [7:0] for Host-Bus 8 accesses or bits [15:0] for Host-Bus 16 accesses. When the WORD bit is set, the EPI controller can automatically route bytes of data onto the correct byte lanes such that bytes or words of data can be transferred on the correct byte or half-word bits on the entire bus. For example, the most significant byte of data will be transferred on bits [31:28] in host-bus 8 mode and the most significant word of data will be transferred on bits [31:16] of Host-Bus 16 mode. In addition, for the three modes above (1, 2, 4) that the Host-Bus 16 mode supports, byte select signals can be optionally implemented by setting the BSEL bit in the EPIHB16CFG register.

Note: Byte accesses should not be attempted if the BSEL bit has not been enabled in Host-Bus 16 mode.

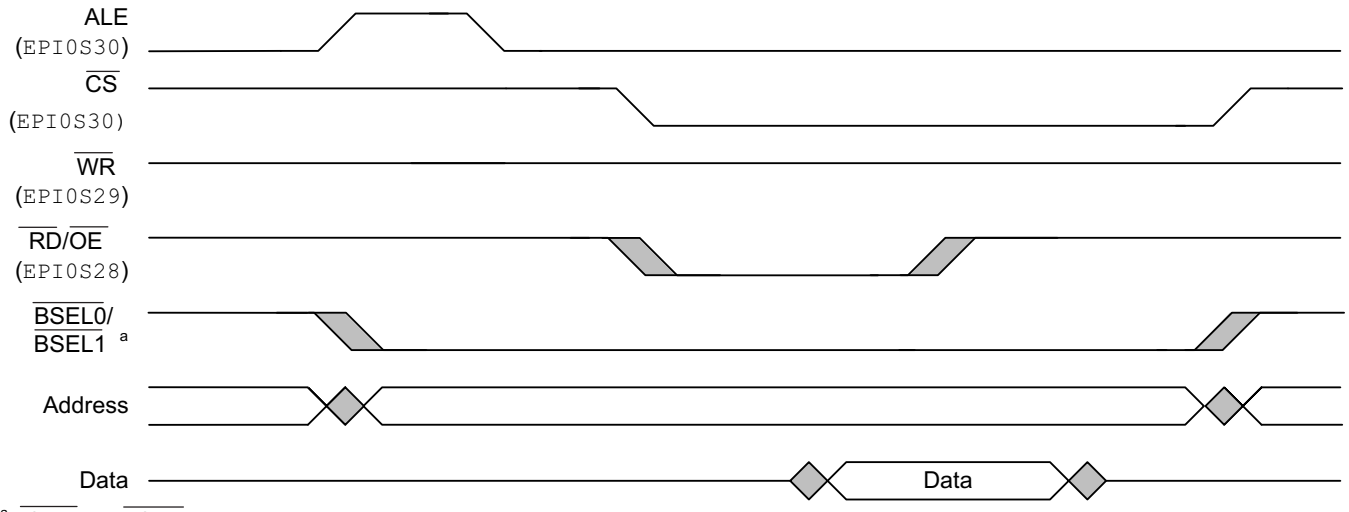
See your device-specific data manual's *EPI Electricals* section for timing details for the Host-Bus mode.

17.7.4 Host Bus Operation

Bus operation is the same in Host-Bus 8 and Host-Bus 16 modes and is asynchronous. Timing diagrams show both ALE and $\overline{CS}$ operation. The optional HB16 byte select signals have the same timing as the address signals. If wait states are required in the bus access, they can be inserted during the data phase of the access using the WRWS and RDWS bits in the EPIHBnCFG2 register. Each wait state adds 2 EPI clock cycles to the duration of the $\overline{WR}$ or $\overline{RD}$ strobe. During idle cycles, the address and muxed address data signals maintain the state of the last cycle.

Figure 17-8 shows a basic Host-Bus read cycle. Figure 17-9 shows a basic Host-Bus write cycle. Both of these figures show address and data signals in the non-multiplexed mode (MODE field ix 0x1 in the EPIHBnCFG register).

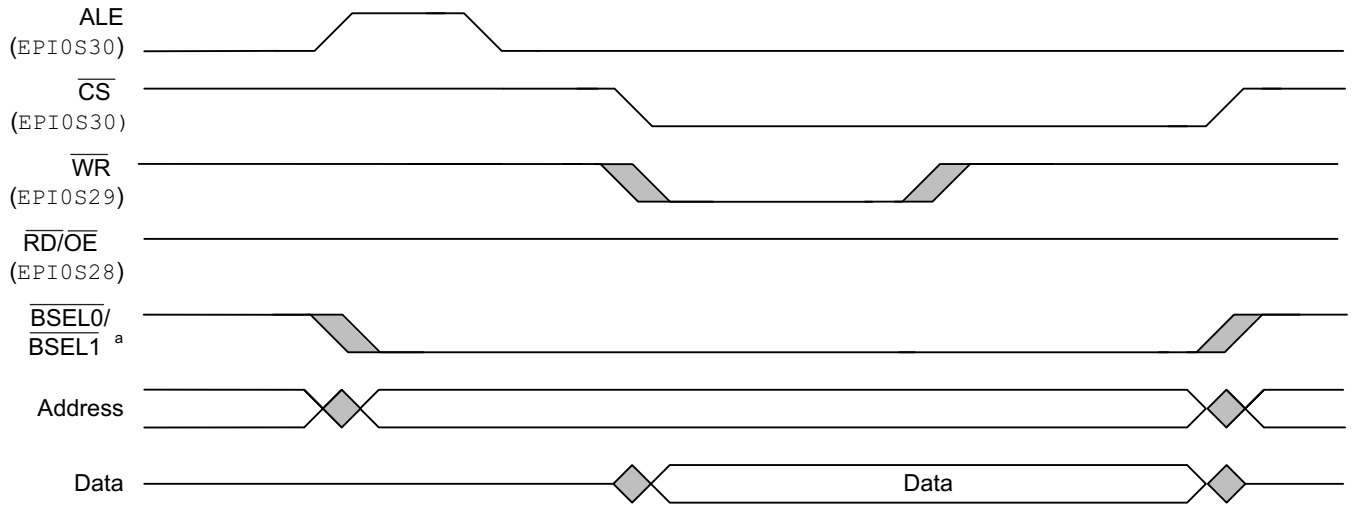
Figure 17-8. Host-Bus Read Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0, ALEHIGH = 1



[a] $\overline{BSEL0}$ and $\overline{BSEL1}$ are available in Host-Bus16 mode only.

Copyright © 2012–2019, Texas Instruments Incorporated

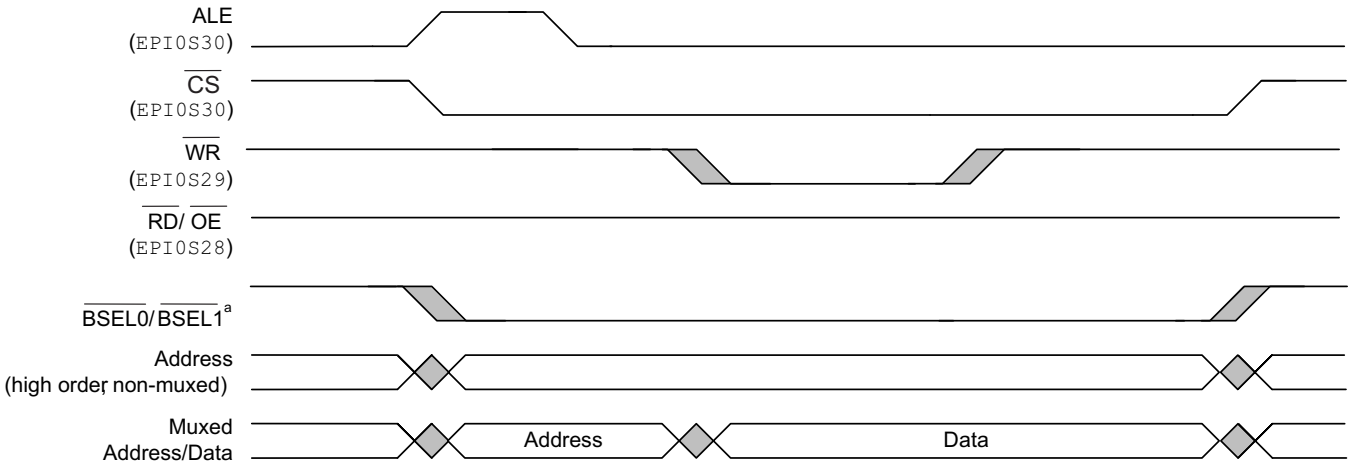
Figure 17-9. Host-Bus Write Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0, ALEHIGH = 1



[a] BSEL0 and BSEL1 are available in Host-Bus16 mode only.

Figure 17-10 shows a write cycle with the address and data signals multiplexed (MODE field is 0x0 in the EPIHBnCFG register). A read cycle would look similar, with the RD strobe being asserted along with CS and data being latched on the rising edge of RD.

Figure 17-10. Host-Bus Write Cycle with Multiplexed Address and Data, MODE = 0x0, WRHIGH = 0, RDHIGH = 0, ALEHIGH = 1



[a] BSEL0 and BSEL1 are available in Host-Bus16 mode only.

When using ALE with dual CS configuration (CSCFGEXT bit is 0 and the CSCFG field is 0x3 in the EPIHBnCFG2 register) or quad chip select (CSCFGEXT bit is 1 and CSCSFG is 0x2), the appropriate CS signal is asserted at the same time as ALE, as shown in Figure 17-11.

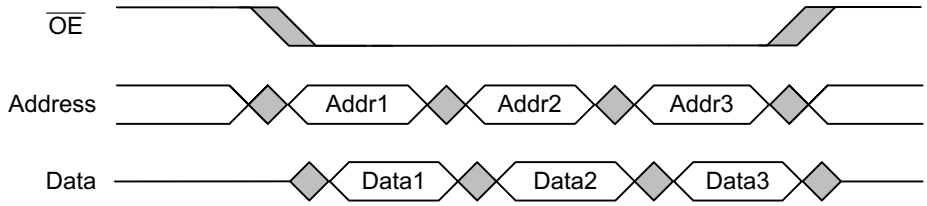
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 17-11. Host-Bus Write Cycle with Multiplexed Address and Data and ALE with Dual or Quad $\overline{CS}$

[a] $\overline{BSEL0}$ and $\overline{BSEL1}$ are available in Host-Bus16 mode only.

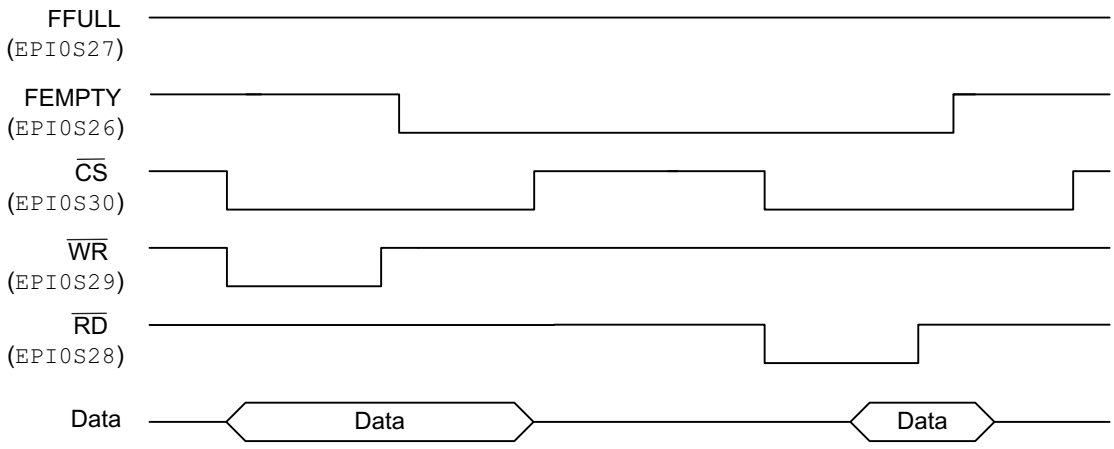
Figure 17-12 shows continuous read mode accesses. In this mode, reads are performed by keeping the read mode selected (output enable is asserted) and then changing the address pins. The data pins are changed by the SRAM after the address pins change.

Figure 17-12. Continuous Read Mode Accesses



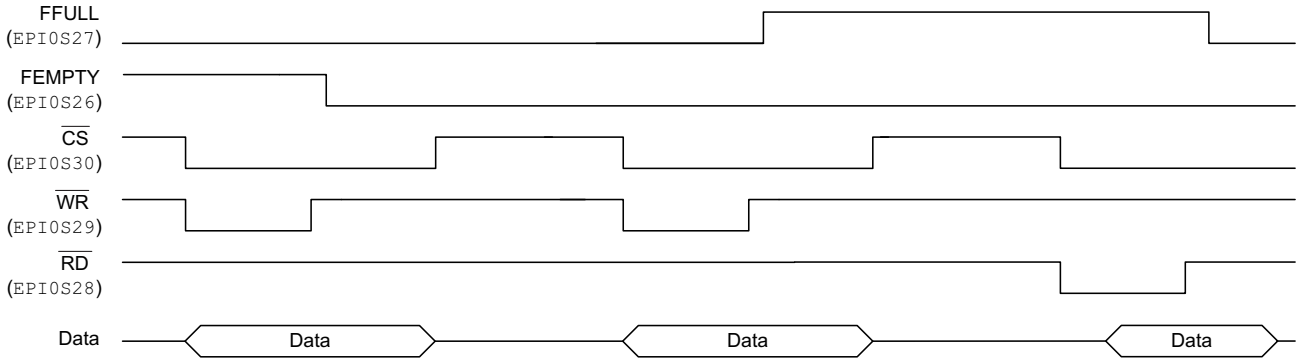
FIFO mode accesses are the same as normal read and write accesses, except that the ALE signal and address pins are not present. Two input signals can be used to indicate when the XFIFO is full or empty to gate transactions and avoid overruns and underruns. The FFULL and FEMPTY signals are synchronized and must be recognized as asserted by the microcontroller for two system clocks before they affect transaction status. The MAXWAIT field in the EPIHBnCFG register defines the maximum number of EPI clocks to wait while the FEMPTY or FFULL signal is holding off a transaction. Figure 17-13 shows how the FEMPTY signal should respond to a write and read from the XFIFO. Figure 17-14 shows how the FEMPTY and FFULL signals should respond to two writes and one read from an external FIFO that contains two entries.

Figure 17-13. Write Followed by Read to External FIFO



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 17-14. Two-Entry FIFO



17.8 General-Purpose Mode

The General-Purpose Mode Configuration (EPIGPCFG) register is used to configure the control, data, and address pins, if used. Any unused EPI controller signals can be used as GPIOs or another alternate function. The general-purpose configuration can be used for custom interfaces with FPGAs, CPLDs, and digital data acquisition and actuator control.

NOTE: The RD2CYC bit in the EPIGPCFG register must be set at all times in General-Purpose mode to ensure proper operation.

General-Purpose mode is designed for three general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs. Three sizes of data and optional address are supported. Framing and clock-enable functions permit more optimized interfaces.
- General parallel GPIO. From 1 to 32 pins may be written or read, with the speed precisely controlled by the EPIBAUD register baud rate (when used with the WFIFO and/or the NBRFIFO) or by the rate of accesses from software or µDMA. Examples of this type of use include:
  - Reading 20 sensors at fixed time periods by configuring 20 pins to be inputs, configuring the COUNT0 field in the EPIBAUD register to some divider, and then using non-blocking reads.
  - Implementing a very wide ganged PWM/PCM with fixed frequency for driving actuators, LEDs, etc.
  - Implementing SDIO 4-bit mode where commands are driven or captured on six pins with fixed timing, fed by the µDMA.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free-running or gated), a framing signal (with frame size), a ready input (to stretch transactions), an address (of varying sizes), and data (of varying sizes). Additionally, provisions are made for separating data and address phases.

The interface has the following optional features:

- Use of the EPI clock output is controlled by the CLKPIN bit in the EPIGPCFG register. Unclocked uses include general-purpose I/O and asynchronous interfaces (optionally using RD and WR strobes). Clocked interfaces allow for higher speeds and are much easier to connect to FPGAs and CPLDs (which usually include input clocks).
- EPI clock, if used, may be free running or gated depending on the CLKGATE bit in the EPIGPCFG register. A free-running EPI clock requires another method for determining when data is live, such as the frame pin or RD/WR strobes. A gated clock approach uses a setup-time model in which the EPI clock controls when transactions are starting and stopping. The gated clock is held high until a new transaction is started and goes high at the end of the cycle where RD/WR/FRAME and address (and data if write) are emitted.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: If general purpose mode is configured to 32-bit data transfers (DSIZE = 0x3 in the EPICFG register), CLKPIN must be set to 0 to prevent clock-gated reads.

- Use of the ready input (iRDY) from the external device is controlled by the RDYEN bit in the EPIGPCFG register. The iRDY signal uses EPI0S27 and may only be used with a free-running clock. iRDY gates transactions, no matter what state they are in. When iRDY is deasserted, the transaction is held off from completing.
- Use of the frame output (FRAME) is controlled by the FRMPIN bit in the EPIGPCFG register. The frame pin maybe used whether the clock is output or not, and whether the clock is free running or not. It may also be used along with the iRDY signal. The frame may be a pulse (one clock) or may be 50/50 split across the frame size (controlled by the FRM50 bit in the EPIGPCFG register). The frame count (the size of the frame as specified by the FRMCNT field in the EPIGPCFG register) may be between 1 and 15 clocks for pulsed and between 2 and 30 clocks for 50/50. The frame pin counts transactions and not clocks; a transaction is any clock where the RD or WR strobe is high (if used). So, if the FRMCNT bit is set, then the frame pin pulses every other transaction; if 2-cycle reads and writes are used, it pulses every other address phase. FRM50 must be used with this in mind as it may hold state for many clocks waiting for the next transaction.
- Use of the RD and WR outputs is controlled by the RW bit in the EPIGPCFG register. For interfaces where the direction is known (in advance, related to frame size, or other means), these strobes are not needed. For most other interfaces, RD and WR are used so the external peripheral knows what transaction is taking place, and if any transaction is taking place.
- Separation of address/request and data phases may be used on writes using the WR2CYC bit in the EPIGPCFG register. This configuration allows the external peripheral extra time to act. Address and data phases must be separated on reads, and the RD2CYC bit in the EPIGPCFG register must be set. When configured to use an address as specified by the ASIZE field in the EPIGPCFG register, the address is emitted on the with the RD strobe (first cycle) and data is expected to be returned on the next cycle (when RD is not asserted). If no address is used, then RD is asserted on the first cycle and data is captured on the second cycle (when RD is not asserted), allowing more setup time for data. For writes, the output may be in one or two cycles. In the two-cycle case, the address (if any) is emitted on the first cycle with the WR strobe and the data is emitted on the second cycle (with WR not asserted). Although split address and write data phases are not normally needed for logic reasons, it may be useful to make read and write timings match. If 2-cycle reads or writes are used, the RW bit is automatically set.
- Address may be emitted (controlled by the ASIZE field in the EPIGPCFG register). The address may be up to 4 bits (16 possible values), up to 12 bits (4096 possible values), or up to 20 bits (1 M possible values). Size of address limits size of data, for example, 4 bits of address support up to 24 bits data. 4-bit address uses EPI0S[27:24]; 12-bit address uses EPI0S[27:16]; 20-bit address uses EPI0S[27:8]. The address signals may be used by the external peripheral as an address, code (command), or for other unrelated uses (such as a chip enable). If the chosen address/data combination does not use all of the EPI signals, the unused pins can be used as GPIOs or for other functions. For example, when using a 4-bit address with an 8-bit data, the pins assigned to EPIS0[23:8] can be assigned to other functions.
- Data may be 8 bits, 16 bits, 24 bits, or 32 bits (controlled by the DSIZE field in the EPIGPCFG register).32-bit data cannot be used with address or EPI clock or any other signal. 24-bit data can only be used with 4-bit address or no address. 32-bit data requires that either the WR2CYC bit or the RD2CYC bit in the EPIGPCFG register is set.
- Memory can be used more efficiently by using the Word Access Mode. By default, the EPI controller uses data bits [7:0] when the DSIZE field in the EPIGPCFG register is 0x0; data bits [15:0] when the DSIZE field is 0x1; data bits [23:0] when the DSIZE field is 0x2; and data bits [31:0] when the DSIZE field is 0x3. When the WORD bit in the EPIGPCFG2 register is set, the EPI controller automatically routes bytes of data onto the correct byte lanes such that data can be stored in bits [31:8] for DSIZE=0x0 and bits [31:16] for DSIZE=0x1.
- When using the EPI controller as a GPIO interface, writes are FIFOed (up to 4 can be held at any time), and up to 32 pins are changed using the EPIBAUD clock rate specified by COUNT0. As a result, output pin control can be very precisely controlled as a function of time. By contrast, when writing to normal GPIOs, writes can only occur 8-bits at a time and take up to two clock cycles to complete. In addition, the write itself may be further delayed by the bus due to µDMA or draining of a previous write. With both GPIO and the EPI controller, reads may be performed directly, in which case the current pin

Copyright © 2012–2019, Texas Instruments Incorporated

states are read back. With the EPI controller, the non-blocking interface may also be used to perform reads based on a fixed time rule via the EPIBAUD clock rate.

Table 17-9 shows how the EPI0S[31:0] signals function while in General-Purpose mode. Notice that the address connections vary depending on the data-width restrictions of the external peripheral.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-9. EPI General-Purpose Signal Connections

EPI Signal	General-Purpose Signal (D8, A20)	General-Purpose Signal (D16, A12)	General-Purpose Signal (D24, A4)	General-Purpose Signal (D32)
EPI0S0	D0	D0	D0	D0
EPI0S1	D1	D1	D1	D1
EPI0S2	D2	D2	D2	D2
EPI0S3	D3	D3	D3	D3
EPI0S4	D4	D4	D4	D4
EPI0S5	D5	D5	D5	D5
EPI0S6	D6	D6	D6	D6
EPI0S7	D7	D7	D7	D7
EPI0S8	A0	D8	D8	D8
EPI0S9	A1	D9	D9	D9
EPI0S10	A2	D10	D10	D10
EPI0S11	A3	D11	D11	D11
EPI0S12	A4	D12	D12	D12
EPI0S13	A5	D13	D13	D13
EPI0S14	A6	D14	D14	D14
EPI0S15	A7	D15	D15	D15
EPI0S16	A8	A0[1]	D16	D16
EPI0S17	A9	A1	D17	D17
EPI0S18	A10	A2	D18	D18
EPI0S19	A11	A3	D19	D19
EPI0S20	A12	A4	D20	D20
EPI0S21	A13	A5	D21	D21
EPI0S22	A14	A6	D22	D22
EPI0S23	A15	A7	D23	D23
EPI0S24	A16	A8	A0[2]	D24
EPI0S25	A17	A9	A1	D25
EPI0S26	A18	A10	A2	D26
EPI0S27	A19/iRDY[3]	A11/iRDY[3]	A3/iRDY[3]	D27
EPI0S28	WR	WR	WR	D28
EPI0S29	RD	RD	RD	D29
EPI0S30	Frame	Frame	Frame	D30
EPI0S31	Clock	Clock	Clock	D31

(1) In this mode, half-word accesses are used. A0 is the LSB of the address and is equivalent to the system A1 address.
(2) In this mode, word accesses are used. A0 is the LSB of the address and is equivalent to the system A2 address.
(3) This signal is iRDY if the RDYEN bit in the EPIGPCFG register is set.

17.8.1 General Purpose Bus Operation

A basic access is one EPI clock for write cycles and two EPI clocks for read cycles. An additional EPI clock can be inserted into a write cycle by setting the WR2CYC bit in the EPIGPCFG register. Note that the RD2CYC bit must always be set in the EPIGPCFG register.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 17-15. Single-Cycle Write Access, FRM50 = 0, FRMCNT = 0, WR2CYC = 0

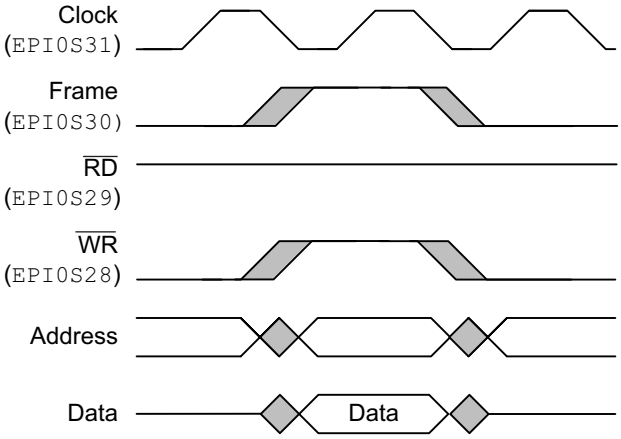


Figure 17-16. Two-Cycle Read, Write Accesses, FRM50 = 0, FRMCNT = 0, RD2CYC = 1, WR2CYC = 1

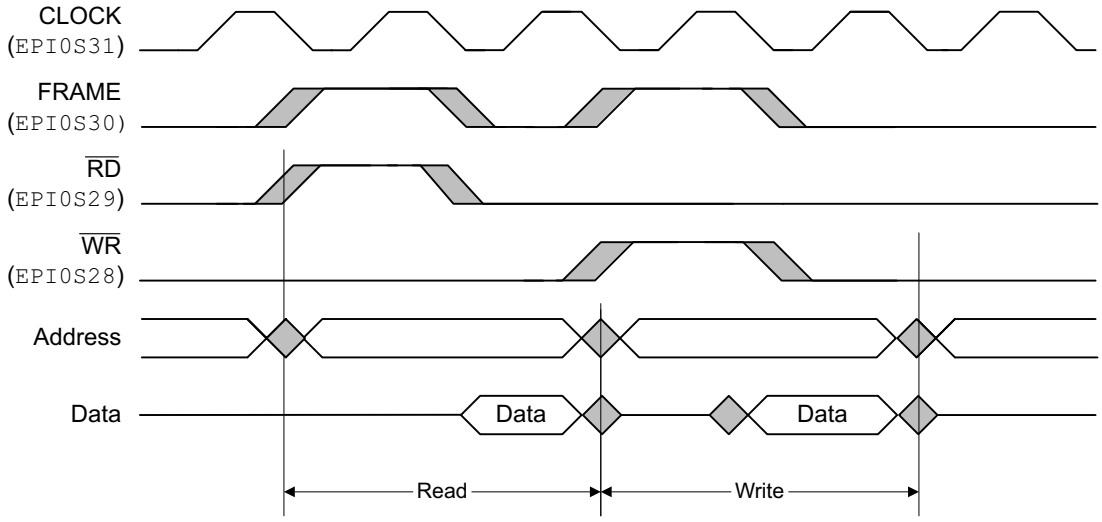
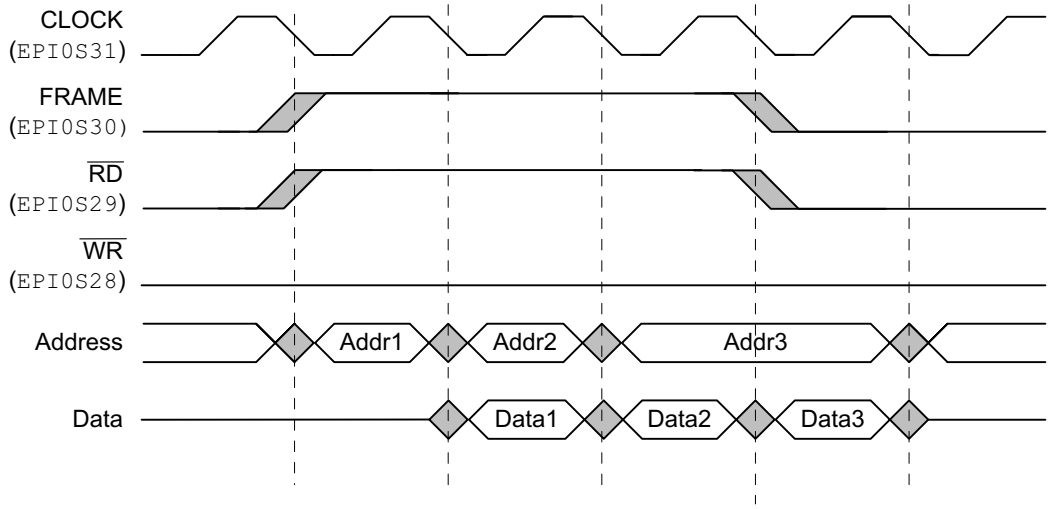


Figure 17-17. Read Accesses, FRM50 = 0, FRMCNT = 0, RD2CYC = 1

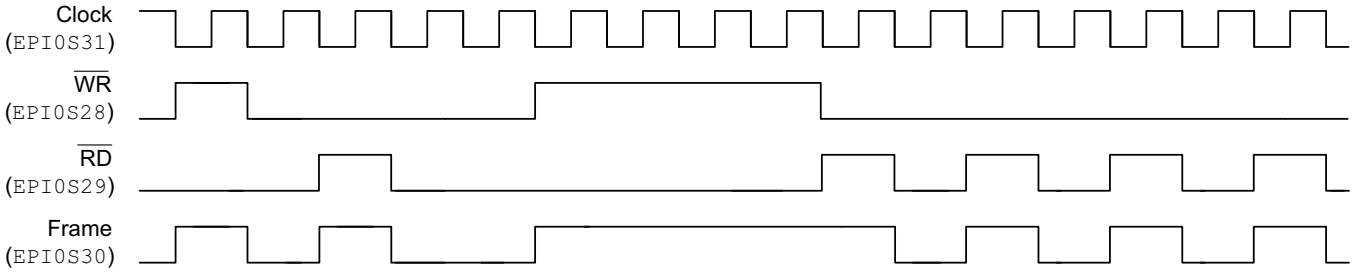


Copyright © 2012–2019, Texas Instruments Incorporated

17.8.1.1 FRAME Signal Operation

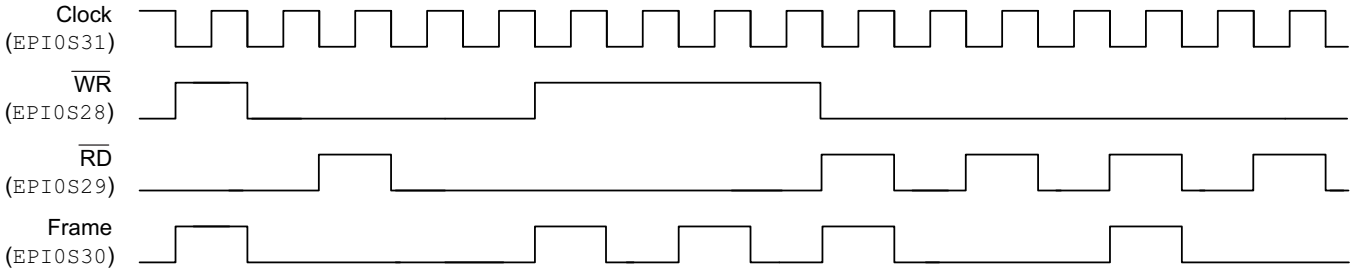
The operation of the FRAME signal is controlled by the FRMCNT and FRM50 bits. When FRM50 is clear, the FRAME signal is high whenever the WR or RD strobe is high. When FRMCNT is clear, the FRAME signal is simply the logical OR of the WR and RD strobes, so the FRAME signal is high during every read or write access. See Figure 17-18.

Figure 17-18. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 0



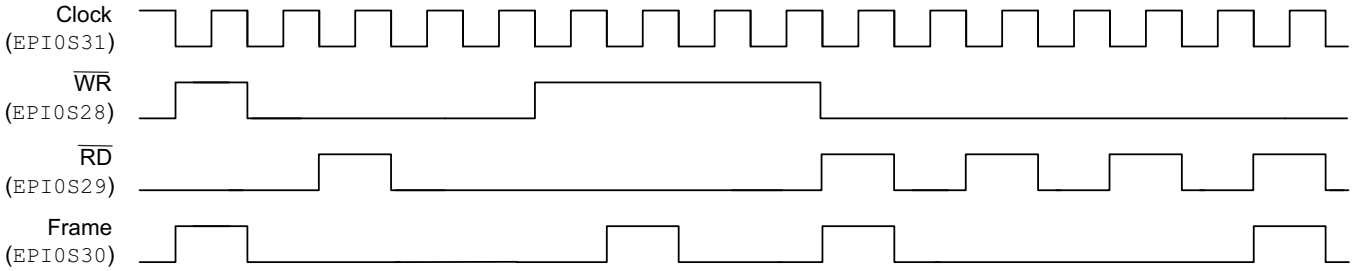
If the FRMCNT field is 0x1, then the FRAME signal pulses high during every other read or write access. See Figure 17-19.

Figure 17-19. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 1



If the FRMCNT field is 0x2 and FRM50 is clear, then the FRAME signal pulses high during every third access, and so on for every value of FRMCNT. See Figure 17-20.

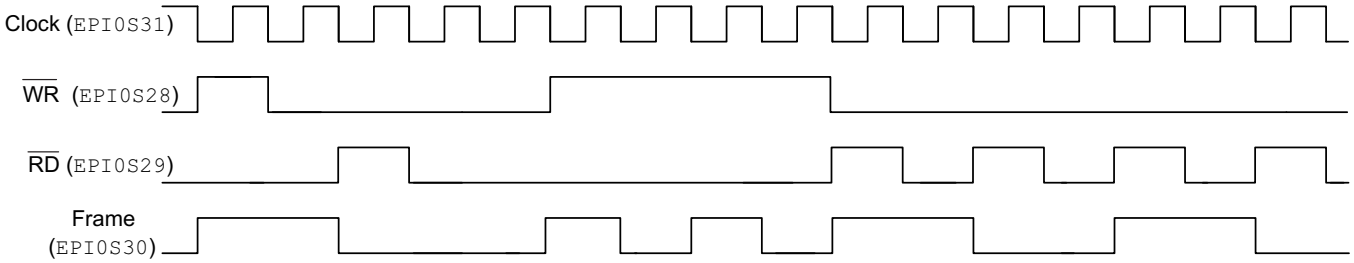
Figure 17-20. FRAME Signal Operation, FRM50 = 0 and FRMCNT = 2



When FRM50 is set, the FRAME signal transitions on the rising edge of either the WR or RD strobes. When FRMCNT=0, the FRAME signal transitions on the rising edge of WR or RD for every access. See Figure 17-21.

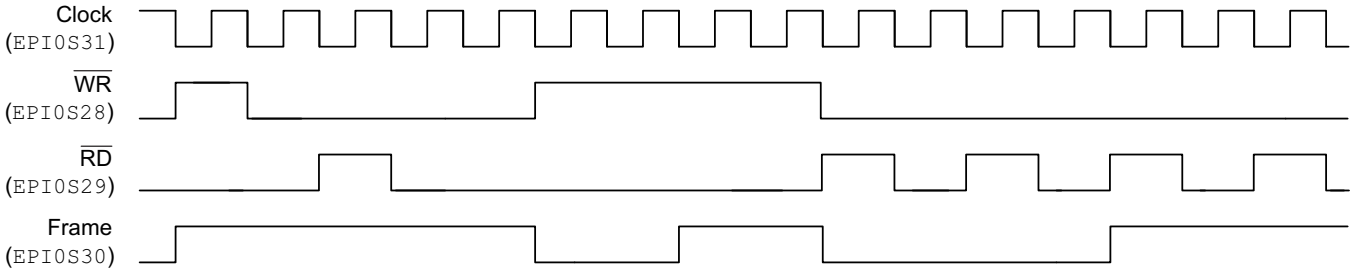
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 17-21. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 0



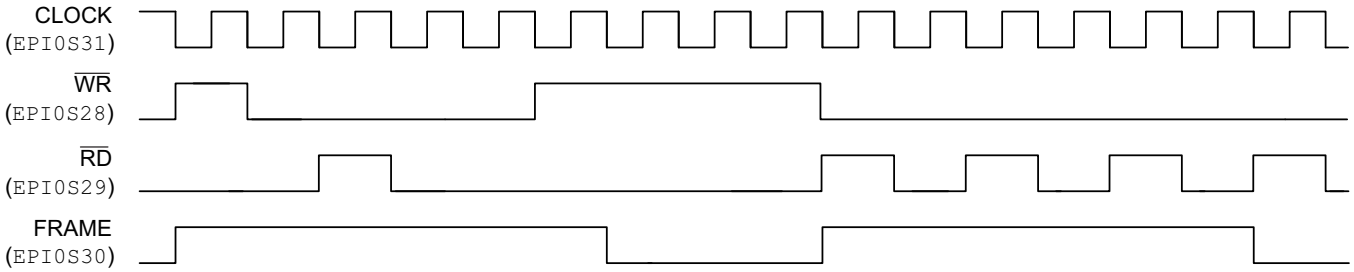
When FRMCNT=1, the FRAME signal transitions on the rising edge of the WR or RD strobes for every other access. See Figure 17-22.

Figure 17-22. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 1



When FRMCNT=2, the FRAME signal transitions the rising edge of the WR or RD strobes for every third access, and so on for every value of FRMCNT. See Figure 17-23.

Figure 17-23. FRAME Signal Operation, FRM50 = 1 and FRMCNT = 2



Copyright © 2012–2019, Texas Instruments Incorporated

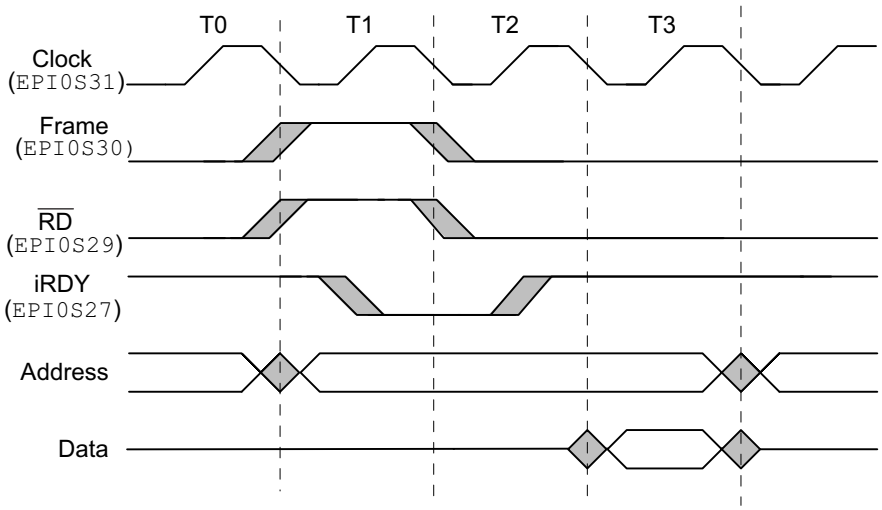
17.8.1.2 iRDY Signal Operation

The ready input (iRDY) signal can be used to lengthen bus cycles and is enabled by the RDYEN bit in the EPIGPCFG register. iRDY is input on EPI0S27 and may only be used with a free-running clock (CLKGATE is clear). If iRDY is deasserted, further transactions are held off until the iRDY signal is asserted again. iRDY is sampled on the falling edge of the EPI clock and gates transactions, no matter what state they are in

A two-cycle access has two phases in the bus cycle. The first clock is the address phase, and the second clock is the data phase. If iRDY is sampled Low at the start of the address phase then the address phase is extended (FRAME, RD, and Address are all asserted) until after iRDY has been sampled High again. Data is sampled on the subsequent rising edge.

If iRDY is sampled Low at the start of the data phase, as shown in Figure 17-24, the FRAME, RD, Address, and Data signals behave as they would during a normal transaction in T1. The data phase (T2) is extended with only Address being asserted until iRDY is recognized as asserted again. Data is latched on the subsequent rising edge

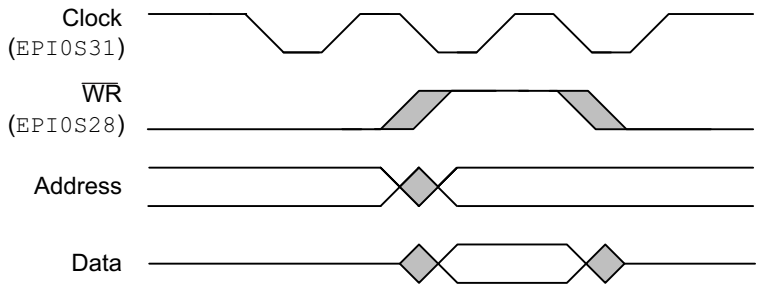
Figure 17-24. iRDY Signal Operation, FRM50 = 0, FRMCNT = 0, and RD2CYC = 1



17.8.1.3 EPI Clock Operation

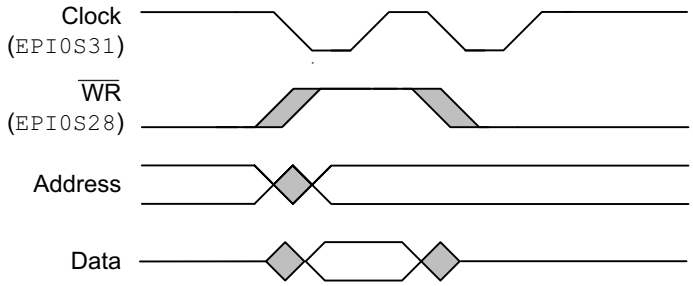
If the CLKGATE bit in the EPIGPCFG register is clear, the EPI clock always toggles when General-purpose mode is enabled. If CLKGATE is set, the clock is output only when a transaction is occurring, otherwise the clock is held high. If the WR2CYC bit is clear, the EPI clock begins toggling 1 cycle before the WR strobe goes high. If the WR2CYC bit is set, the EPI clock begins toggling when the WR strobe goes high. The clock stops toggling after the first rising edge after the WR strobe is deasserted. The RD strobe operates in the same manner as the WR strobe when the WR2CYC bit is set, as the RD2CYC bit must always be set. See Figure 17-25 and Figure 17-26.

Figure 17-25. EPI Clock Operation, CLKGATE = 1, WR2CYC = 0



Copyright © 2012–2019, Texas Instruments Incorporated

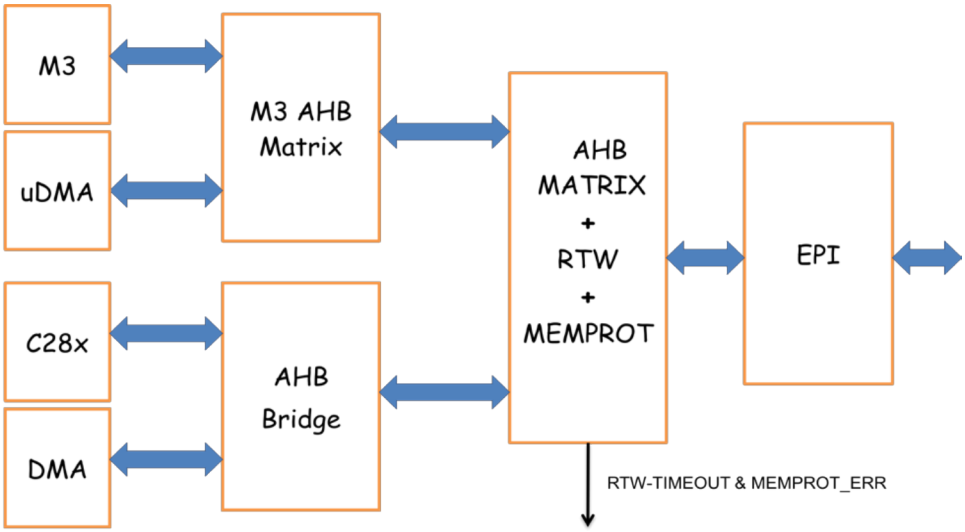
Figure 17-26. EPI Clock Operation, CLKGATE = 1, WR2CYC = 1



17.9 C28x Access to EPI

The EPI peripheral is a shared resource between master subsystem (M3 and µDMA) and control subsystem (C28x and DMA). All masters can access external devices via EPI simultaneously. Access to each master is granted based on round-robin arbitration. EPI configuration registers are shared for READ only between all masters. Only the master subsystem has write access to EPI configuration registers. This means that M3 software must configure the EPI in a particular mode before C28x software can access the external device via EPI.

Figure 17-27. C28x Master and Control Subsystem Access to EPI



Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-10 shows the address mapping for the control subsystem.

Table 17-10. Control Subsystem Address Mapping

Chip Select	C28x/DMA Address Start (16-bit word address)	C28x/DMA Address End (16-bit word address)	Available address lines for control subsystem
$\overline{CS0}$	0x30 0000	0x33 FFFF	A0 to A17
$\overline{CS1}$	0x34 0000	0x37 FFFF	A0 to A17
$\overline{CS2}$	0x38 0000	0x3B FFFF	A0 to A17
$\overline{CS3}$	0x3C 0000	0x3F 7FFF	A0 to A17

17.9.1 Real-Time Window (RTW)

For allowing uninterrupted accesses to C28x-CPU/C28x-DMA for control-loop data transfer, a real-time window (RTW) feature has been provided. By enabling this feature, C28x-CPU/C28x-DMA gets dedicated access of EPI for a specific time period. This feature is enabled by setting the “GRAB” bit in the CEPIRTWCFG register. Once EPI is grabbed by C28x software (GRAB = 1), all accesses to EPI from M3/uDMA will be stalled until EPI is released (GRAB = 0) by C28x software.

To prevent permanently locking the EPI by C28x software (in case of error situation) a timeout feature is provided. User needs to program real time window period register (CEPIRTWPRD) with time period for which C28x wants to grab the EPI. Once C28x grabs the EPI by setting the GRAB bit, a free running counter, CEPIRTWCNT, starts incrementing. If C28x software does not release the EPI access (by setting GRAB bit to ‘0’) before the counter value becomes equal to the value programmed in CEPIRTWPRD register, TIMEOUT happens and GRAB bit gets reset to ‘0’. When TIMEOUT occurs, a flag gets set in CEPIESTATUS register and an interrupt gets generated to the C28x core.

NOTE: The RTW feature is only needed in an application where both subsystems (master and control) are accessing EPI and one needs to provide a window for dedicated access to EPI from the control (C28x) subsystem. Otherwise, by default, both subsystems have access to EPI with round-robin arbitration.

17.9.2 Memory Protection

Since access to external memory space via EPI is shared between all masters, it is important to protect critical data from getting corrupted by unwanted masters. To achieve this, memory protection feature is provided for each $\overline{CSx}$ region. Using this feature, users can enable/disable write accesses to individual $\overline{CSx}$ regions from the master or control subsystem. This is done by programming the MEMPROT register. This register is only accessible by M3 software. By default, accesses to all $\overline{CSx}$ regions are enabled from both subsystems.

NOTE: The memory protection feature is only for enabling or disabling the write access. Read accesses are always enabled for both subsystems.

Copyright © 2012–2019, Texas Instruments Incorporated

17.10 Register Map

Table 17-11 lists the EPI registers. The offset listed is a hexadecimal increment to the register's address, relative to the base address of 0x400D.0000 (ending address of 0x400D.0FFF). Note that the EPI controller clock must be enabled before the registers can be programmed. There must be a delay of three system clocks after the EPI module clock is enabled before any EPI module registers are accessed.

NOTE: A back-to-back write followed by a read of the same register reads the value that written by the first write access, not the value from the second write access. (This situation only occurs when the processor core attempts this action, the µDMA does not do this.) To read back what was just written, another instruction must be generated between the write and read. Read-write does not have this issue, so use of read-write for clear of error interrupt cause is not affected.

For all versions of EPI, only WORD read and write accesses to registers are supported.

Table 17-11. External Peripheral Interface (EPI) Register Map
M3 Base Address: 0x400D_0000, C28x Base Address: 0x7C00

M3 Offset Address	C28x Offset Address	Acronym	Type	Reset	Description
0x000	0x000	EPICFG	R/W	0x0000.0000	EPI Configuration Register
0x004	0x002	EPIBAUD	R/W	0x0000.0000	EPI Main Baud Rate Register
0x010	0x008	EPISDRAMCFG	R/W	0x42EE.0000	EPI SDRAM Configuration Register
0x010	0x008	EPIHB8CFG	R/W	0x0000.0000	EPI Host-Bus 8 Configuration Register
0x010	0x008	EPIHB16CFG	R/W	0x0000.0000	EPI Host-Bus 16 Configuration Register
0x010	0x008	EPIGPCFG	R/W	0x0000.0000	EPI General-Purpose Configuration Register
0x014	0x00A	EPIHB8CFG2	R/W	0x0000.0000	EPI Host-Bus 8 Configuration 2 Register
0x014	0x00A	EPIHB16CFG2	R/W	0x0000.0000	EPI Host-Bus 16 Configuration 2 Register
0x014	0x00A	EPIGPCFG2	R/W	0x0000.0000	EPI General-Purpose Configuration 2 Register
0x01C	0x00E	EPIADDRMAP	R/W	0x0000.0000	EPI Address Map Register
0x020	0x010	EPIRSIZE0	R/W	0x0000.0000	EPI Read Size 0 Register
0x024	0x012	EPIRADDR0	R/W	0x0000.0000	EPI Read Address 0 Register
0x028	0x014	EPIRPSTD0	R/W	0x0000.0000	EPI Non-Blocking Read Data 0 Register
0x030	0x018	EPIRSIZE1	R/W	0x0000.0003	EPI Read Size 1 Register
0x034	0x01A	EPIRADDR1	R/W	0x0000.0000	EPI Read Address 1 Register
0x038	0x01C	EPIRPSTD1	R/W	0x0000.0000	EPI Non-Blocking Read Data 1 Register
0x060	0x030	EPISTAT	R	0x0000.0000	EPI Status Register
0x06C	0x036	EPIRFIFOCNT	R	-	EPI Read FIFO Count Register
0x070	0x038	EPIREADFIFO	R	-	EPI Read FIFO Register
0x074	0x03A	EPIREADFIFO1	R	-	EPI Read FIFO Alias 1 Register
0x078	0x03C	EPIREADFIFO2	R	-	EPI Read FIFO Alias 2 Register

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-11. External Peripheral Interface (EPI) Register Map
M3 Base Address: 0x400D_0000, C28x Base Address: 0x7C00 (continued)

M3 Offset Address	C28x Offset Address	Acronym	Type	Reset	Description
0x07C	0x03E	EPIREADFIFO3	R	-	EPI Read FIFO Alias 3 Register
0x080	0x040	EPIREADFIFO4	R	-	EPI Read FIFO Alias 4 Register
0x084	0x042	EPIREADFIFO5	R	-	EPI Read FIFO Alias 5 Register
0x088	0x044	EPIREADFIFO6	R	-	EPI Read FIFO Alias 6 Register
0x08C	0x046	EPIREADFIFO7	R	-	EPI Read FIFO Alias 7 Register
0x200	0x100	EPIFIFOLVL	R/W	0x0000.0033	EPI FIFO Level Selects Register
0x204	0x102	EPIWFIFOCNT	R	0x0000.0004	EPI Write FIFO Count Register
0x210	0x108	EPIIM	R/W	0x0000.0000	EPI Interrupt Mask Register
0x214	0x10A	EPIRIS	R	0x0000.0004	EPI Raw Interrupt Status Register
0x218	0x10C	EPIMIS	R	0x0000.0000	EPI Masked Interrupt Status Register
0x21C	0x10E	EPIEISC	R/W-1C	0x0000.0000	EPI Error Interrupt Status and Clear Register
0x308	0x184	EPIHB8CFG3	R/W	0x0008.0000	EPI Host-Bus 8 Configuration 3
0x308	0x184	EPIHB16CFG3	R/W		EPI Host-Bus 16 Configuration 3
0x30C	0x186	EPIHB8CFG4	R/W		EPI Host-Bus 8 Configuration 4
0x30C	0x186	EPIHB16CFG4	R/W		EPI Host-Bus 16 Configuration 4
0x310	0x188	EPIHB8TIME	R/W		EPI Host-Bus 8 Timing Extension
0x310	0x188	EPIHB16TIME	R/W		EPI Host-Bus 16 Timing Extension
0x314	0x18A	EPIHB8TIME2	R/W		EPI Host-Bus 8 Timing Extension
0x314	0x18A	EPIHB16TIME2	R/W		EPI Host-Bus 16 Timing Extension
0x318	0x18C	EPIHB8TIME3	R/W		EPI Host-Bus 8 Timing Extension
0x318	0x18C	EPIHB16TIME3	R/W		EPI Host-Bus 16 Timing Extension
0x31C	0x18E	EPIHB8TIME4	R/W		EPI Host-Bus 8 Timing Extension
0x31C	0x18E	EPIHB16TIME4	R/W		EPI Host-Bus 16 Timing Extension

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-12. C28x Base Address: 0x4430

Register Name	Description	Size (x8)	Offset (x16)	Protection
CEPIRTWCFG	C28 EPI Real-Time Window Config Register	4	0x00	EALLOW
CEPIRTWCNT	C28 EPI Real-Time Window Counter Register	4	0x02	None
CEPIRTWPRD	C28 EPI Real-Time Window Period Register	4	0x04	EALLOW
Reserved	Reserved	8	0x06	None
CEPISTATUS	C28 EPI Status Register	4	0x0A	None

Table 17-13. Base Address 0x400F_ B930

Register Name	Description	Size (x8)	Offset (x8)	Protection
MEMPROT	Memory Protection Register	4	0x8	PROTECTED

Copyright © 2012–2019, Texas Instruments Incorporated

17.11 Register Descriptions

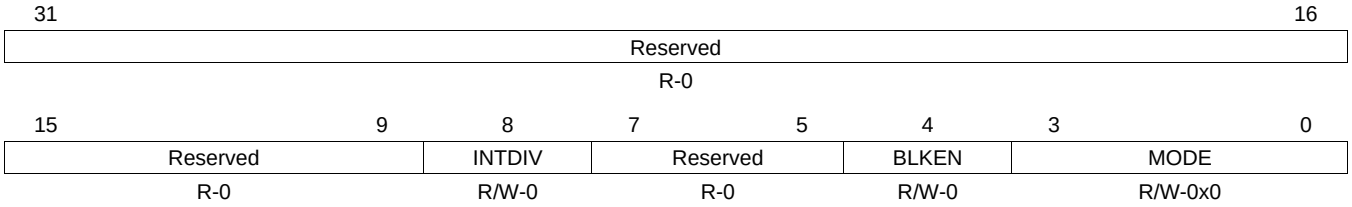
This section lists and describes the EPI registers, in numerical order by address offset.

17.11.1 EPI Configuration Register (EPICFG), offset 0x000

NOTE: The MODE field determines which configuration register is accessed for offsets 0x010 and 0x014. Any write to the EPICFG register resets the register contents at offsets 0x010 and 0x014.

The configuration register is used to enable the block, select a mode, and select the basic pin use (based on the mode). Note that attempting to program an undefined MODE field clears the BLKEN bit and disables the EPI controller.

Figure 17-28. EPI Configuration Register (EPICFG) [offset 0x000]



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-14. EPI Configuration Register (EPICFG) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	INTDIV		Integer Clock Divider Enable
		0	EPIBAUD register values create formula clock divide.
		1	EPIBAUD register values create integer clock divide.
7-5	Reserved		Reserved
4	BLKEN		Block Enable
		0	The EPI controller is disabled.
		1	The EPI controller is enabled.
3-0	MODE		Mode Select
		0x0	General Purpose General-Purpose mode. Control, address, and data pins are configured using the EPIGPCFG and EPIGPCFG2 registers.
		0x1	SDRAM Supports SDR SDRAM. Control, address, and data pins are configured using the EPISDRAMCFG register.
		0x2	8-Bit Host-Bus (HB8) Host-bus 8-bit interface (also known as the MCU interface). Control, address, and data pins are configured using the EPIHB8CFG and EPIHB8CFG2 registers.
		0x3	16-Bit Host-Bus (HB16) Host-bus 16-bit interface (standard SRAM). Control, address, and data pins are configured using the EPIHB16CFG and EPIHB16CFG2 registers.
		0x3-0xF	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.2 EPI Main Baud Rate (EPIBAUD) Register, offset 0x004

The system clock is used internally to the EPI controller. The baud rate counter can be used to divide the system clock down to control the speed on the external interface. If the mode selected emits an external EPI clock, this register defines the EPI clock emitted. If the mode selected does not use an EPI clock, this register controls the speed of changes on the external interface. This register must be programmed properly so that the speed of the external bus corresponds to the speed of the external peripheral and puts acceptable current load on the pins. COUNT0 is the bit field used in all modes except in HB8 and HB16 modes with dual chip selects and quad chip selects when different baud rates are selected. See Section 17.11.8 and Section 17.11.9. If different baud rates are used, COUNT0 is associated with the address range specified by $\overline{CS1}$. The EPIBAUD2 register configures the baud rates for $\overline{CS2}$ and $\overline{CS3}$.

The COUNTn field is not a straight divider or count. The EPI Clock on EPI0S31 is related to the COUNTn field and the system clock as follows:

If COUNTn = 0,
EPIClockFreq = *SystemClockFreq*

otherwise

EPIClockFreq = *SystemClockFreq* / ([Countn / 2] + 1) × 2

where the symbol around COUNTn/2 is the floor operator; meaning the largest integer less than or equal to COUNTn/2.

So, for example, a COUNTn of 0x0001 results in a clockrate of 1/2 (system clock); a COUNTn of 0x0002 or 0x0003 results in a clock rate of 1/4 (system clock).

The baud rate counter can also be configured as an integer divide by enabling INTDIV in the EPICFG register. When enabled, COUNTn of 0x0000 or 0x0001 results in a clock rate equal to system clock. COUNTn of 0x0002 results in a clock rate of 1/2 (system clock). COUNTn of 0x0003 results in a clock rate of 1/3 (system clock).

Figure 17-29. EPI Main Baud Rate (EPIBAUD) Register [offset 0x004]

31	16	15	0
COUNT1		COUNT0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-15. EPI Main Baud Rate (EPIBAUD) Register Field Descriptions

Bit	Field	Value	Description
31-16	COUNT1		Baud Rate Counter 1 This bit field is only valid with multiple chip selects which are enabled when the CSCFG field is 0x2 or 0x3 or the CSCFGEXT field is set to 1, with CSCFG field as 0x1 or 0x2 and the CSBAUD bit is set in the EPIHBnCFG2 register.
		0	A count of 0 means the system clock is used as is.
		1	This bit field contains a counter used to divide the system clock by the count.
15-0	COUNT0		Baud Rate Counter 0
		0	A count of 0 means the system clock is used as is.
		1	This bit field contains a counter used to divide the system clock by the count.

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.3 EPI Main Baud Rate (EPIBAUD2) Register, offset 0x008

The system clock is used internally to the EPI Controller. The baud rate counter can be used to divide the system clock down to control the speed on the external interface. If the mode selected emits an external EPI clock, this register defines the EPI clock emitted. If the mode selected does not use an EPI clock, this register controls the speed of changes on the external interface. This register must be programmed properly so that the speed of the external bus corresponds to the speed of the external peripheral and puts acceptable current load on the pins. COUNT0 and COUNT1 are used in quad chip select mode when different baud rates are selected, See Section 17.11.8 and Section 17.11.9. If different baud rates are used, COUNT0 is associated with the address range specified by $\overline{CS2}$ and COUNT1 is associated with the address range specified by $\overline{CS3}$.

The COUNTn field is not a straight divider or count. The EPI Clock on EPI0S31 is related to the COUNTn field and the system clock as follows:

If COUNTn = 0,
EPIClockFreq = *SystemClockFreq*

otherwise

EPIClockFreq = *SystemClockFreq* / ([Countn / 2] + 1) × 2

where the symbol around COUNTn/2 is the floor operator; meaning the largest integer less than or equal to COUNTn/2.

So, for example, a COUNTn of 0x0001 results in a clock rate of 1/2 (system clock); a COUNTn of 0x0002 or 0x0003 results in a clock rate of 1/4 (system clock)

Figure 17-30. EPI Main Baud Rate (EPIBAUD2) Register [offset 0x008]

31	16	15	0
COUNT1		COUNT0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-16. EPI Main Baud Rate (EPIBAUD2) Register Field Descriptions

Bit	Field	Value	Description
31-16	COUNT1		$\overline{CS3}$ Baud Rate Counter 1 This bit field contains a counter used to divide the system clock by the count.
		0	A count of 0 means the system clock is used as is.
		1	This bit field is only valid when quad chip selects are enabled by setting the CSCFGEXT bit to 1 and the CSCFG field to 0x1 or 0x2. In addition, the CSBAUD bit must be set in the EPIHBnCFG2 register.
15-0	COUNT0		$\overline{CS2}$ Baud Rate Counter 0 This bit field contains a counter used to divide the system clock by the count.
		0	A count of 0 means the system clock is used as is.
		1	This bit field is only valid when quad chip selects are enabled by setting the CSCFGEXT to 1 and the CSCFG field to 0x1 or 0x2. In addition, the CSBAUD bit must be set in the EPIHBnCFG2 register.

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.4 EPI SDRAM Configuration (EPISDRAMCFG) Register, offset 0x010

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPISDRAMCFG, the MODE field must be 0x1.

The SDRAM Configuration register is used to specify several parameters for the SDRAM controller. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the SDRAM mode is selected again, the values must be reinitialized..

The SDRAM interface designed to interface to x16 SDRS DRAMs of 64 MHz or higher, with the address and data pins overlapped (wire ORed on the board). See Table 17-1 for pin assignments.

Figure 17-31. EPI SDRAM Configuration (EPISDRAMCFG) Register [offset 0x010]

31	30	29	27	26	16
FREQ		Reserved		RFSH	
R/W-0x1		R-0x0		R/W-0x2EE	

15	10	9	8	2	1	0
Reserved		SLEEP	Reserved		SIZE	
R-0		R/W-0	R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-17. EPI SDRAM Configuration (EPISDRAMCFG) Register Field Descriptions

Bit	Field	Value	Description
31-30	FREQ		Frequency Range This field configures the frequency range used for delay references by internal counters. This EPI frequency is the system frequency with the divider programmed by the COUNT0 bit in the EPIBAUDn register bit. This field affects the power up, precharge, and auto refresh delays. This field does not affect the refresh counting, which is configured separately using the RFSH field (and is based on system clock rate and number of rows per bank). The ranges are:
		0x0	0 - 15 MHz
		0x1	15 - 30 MHz
		0x2	30 - 50 MHz
		0x3	50 - 100 MHz
29-27	Reserved		Reserved
26-16	RFSH		Refresh Counter This field contains the refresh counter in system clocks. The reset value of 0x2EE provides a refresh period of 64 ms when using a 50 MHz clock.
15-10	Reserved		Reserved
9	SLEEP		Sleep Mode
		0	No effect
		1	The SDRAM is put into low power state, but is self-refreshed.
8-2	Reserved		Reserved
1-0	SIZE		Size of SDRAM The value of this field affects address pins and behavior.
		0x0	64 megabits (8MB)
		0x1	128 megabits (16MB)
		0x2	256 megabits (32MB)
		0x3	512 megabits (64MB)

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.5 EPI Host-Bus 8 Configuration (EPIHB8CFG) Register, offset 0x010

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIHB8CFG, the MODE field must be 0x2.

The Host Bus 8 Configuration register is activated when the HB8 mode is selected. The HB8 mode supports muxed address/data (overlay of lower 8 address and all 8 data pins), separate address/data, and address-less FIFO mode. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the HB8 mode is selected again, the values must be reinitialized.

This mode is intended to support SRAMs, Flash memory(read), FIFOs, CPLDs/FPGAs, and devices with an MCU/Host Bus slave or 8-bit FIFO interface support.

If less address pins are required, the corresponding AFSEL bit (see the *GPIO* chapter) should not be enabled so the EPI controller does not drive those pins, and they are available as standard GPIOs.

EPI Host-Bus 8 Mode can be configured to use one to four chip selects with and without the use of ALE. If an alternative to chip selects are required, a chip enable can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally $\overline{CE}$, lower addresses are not used. For example, if pins EPI0S27 and EPI0S26 are used for device 1 and 0 respectively, then address 0x6800.0000 accesses device 0 (device 1 has its $\overline{CE}$ high), and 0x6400.0000 accesses device 1 (device 0 has its $\overline{CE}$ high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as CEn because the address remains stable after the ALE strobe. The subsequent $\overline{WR}$ or $\overline{RD}$ signals write or read when ALE is low thus providing $\overline{CE}$ functionality.

Figure 17-32. EPI Host-Bus 8 Configuration (EPIHB8CFG) Register [offset 0x010]

31	30	29	28	27	26	25	24
CLKGATE	CLKGATEI	CLKINV	RDYEN	IRDYINV	Reserved		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0		
23	22	21	20	19	18	17	16
XFFEN	XFEEN	WRHIGH	RDHIGH	ALEHIGH	Reserved		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R-0		
15	14	13	12	11	10	9	8
MAXWAIT							
R/W-1							
7	6	5	4	3	2	1	0
WRWS		RDWS		Reserved		MODE	
R/W-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-18. EPI Host-Bus 8 Configuration (EPIHB8CFG) Register Field Descriptions

Bit	Field	Value	Description
31	CLKGATE		Clock Gated
		0	The EPI clock is free running.
		1	The EPI clock is held low.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-18. EPI Host-Bus 8 Configuration (EPIHB8CFG) Register Field Descriptions (continued)

Bit	Field	Value	Description
30	CLKGATEI		Clock Gated when Idle
		0	The EPI clock is free running
			Note that EPI0S32 is an iRDY signal if RDYEN is set. CLKGATEI is ignored if CLKPIN is 0 or if the COUNT0 field in the EPIBAUD register is cleared.
		1	The EPI clock is output only when there is data to write or read (current transaction); otherwise the EPI clock is held low
29	CLKINV		Invert Output Clock Enable
		0	No effect
		1	Invert EPI clock to ensure the rising edge is centered for outbound signal's setup and hold. Inbound signal is captured on rising edge EPI clock.
28	RDYEN		Input Ready Enable
		0	No effect.
		1	An external ready can be used to control the continuation of the current access. If this bit is set and the iRDY signal (EPIS032) is low, the current access is stalled.
27	IRDYINV		Input Ready Invert
		1	Invert the polarity of incoming external ready (iRDY signal). If this bit is set and the iRDY signal (EPIS032) is high the current access is stalled.
		0	No effect.
26-24	Reserved		Reserved
23	XFFEN		External FIFO FULL Enable
		0	No effect
		1	An external FIFO full signal can be used to control write cycles. If this bit is set and the FFULL full signal is high, XFIFO writes are stalled.
22	XFEEN		External FIFO EMPTY Enable
		0	No effect
		1	An external FIFO empty signal can be used to control read cycles. If this bit is set and the FEMPTY signal is high, XFIFO reads are stalled.
21	WRHIGH		WRITE Strobe Polarity
		0	The WRITE strobe for $\overline{CS0}$ is $\overline{WR}$ (active Low).
		1	The WRITE strobe for $\overline{CS0}$ is WR (active High).
20	RDHIGH		READ Strobe Polarity
		0	The READ strobe for $\overline{CS0}$ is $\overline{RD}$ (active Low).
		1	The READ strobe for $\overline{CS0}$ is RD (active High).
19	ALEHIGH		ALE Strobe Polarity
		0	The address latch strobe for $\overline{CS0}$ accesses is $\overline{ALE}$ (active Low).
		1	The address latch strobe for $\overline{CS0}$ accesses is ALE (active High).
18-16	Reserved		Reserved
15-8	MAXWAIT		Maximum Wait This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY). When the MAXWAIT value is reached the ERRRIS interrupt status bit is set in the EPIRIS register. When this field is clear, the transaction can be held off forever without a system interrupt. **Note:** When the MODE field is configured to be 0x2 and the BLKEN bit is set in the EPICFG register, enabling HB8 mode, this field defaults to 0xFF.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-18. EPI Host-Bus 8 Configuration (EPIHB8CFG) Register Field Descriptions (continued)

Bit	Field	Value	Description
7-6	WRWS		Write Wait States This field adds wait states to the data phase of $\overline{CS0}$ (the address phase is not affected). The effect is to delay the rising edge of $\overline{WR}$ (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the EPIHB8TIME register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is not applicable in BURST mode.
		0x0	Active $\overline{WR}$ is 2 EPI clocks
		0x1	Active $\overline{WR}$ is 4 EPI clocks
		0x2	Active $\overline{WR}$ is 6 EPI clocks
		0x3	Active $\overline{WR}$ is 8 EPI clocks
			This field is used in conjunction with the EPIBAUD register.
5-4	RDWS		Read Wait States This field adds wait states to the data phase (the address phase is not affected). The effect is to delay the rising edge of $\overline{RD}$/Oen (or the falling edge of RD). Each wait state adds two EPI clock cycles to the access time. The RDWSM bit in the EPIHB8TIME register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is not applicable in BURST mode.
		0x0	Active $\overline{RD}$ is 2 EPI clocks.
		0x1	Active $\overline{RD}$ is 4 EPI clocks.
		0x2	Active $\overline{RD}$ is 6 EPI clocks
		0x3	Active $\overline{RD}$ is 8 EPI clocks
			This field is used in conjunction with the EPIBAUD register
3-2	Reserved		Reserved
1-0	MODE		Host Bus Sub-Mode This field determines which of four Host Bus 8 sub-modes to use. Sub-mode use is determined by the connected external peripheral. When used with multiple chip select option and the CSBAUD bit is set to 1 in the EPIHB8CFG2 register, this configuration is for $\overline{CS0}$. If the multiple chip select option is enabled and CSBAUD is clear, all chip-selects use the MODE encoding programmed in this register.
		0x0	ADMUX – AD[7:0] Data and Address are muxed.
		0x1	ADNONMUX – D[7:0] Data and address are separate.
		0x2	Continuous Read - D[7:0] This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of $\overline{OE}$ strobing.
		0x3	XFIFO – D[7:0] This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE.

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.6 EPI Host-Bus 16 Configuration (EPIHB16CFG) Register. offset 0x010

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIHB16CFG, the MODE field must be 0x3.

The Host Bus 16 sub-configuration register is activated when the HB16 mode is selected. The HB16 mode supports muxed address/data (overlay of lower 16 address and all 16 data pins), separated address/data, and address-less FIFO mode. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the HB16 mode is selected again, the values must be reinitialized.

This mode is intended to support SRAMs, Flash memory (read), FIFOs, and CPLDs/FPGAs, and devices with an MCU/Host Bus slave or 16-bit FIFO interface support.

If less address pins are required, the corresponding AFSEL bit (see the GPIOs chapter) should not be enabled so the EPI controller does not drive those pins, and they are available as standard GPIOs.

EPI Host-Bus 16 Mode can be configured to use one to four chip selects with and without the use of ALE. If an alternative to chip selects are required, a chip enable can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally $\overline{CE}$, lower addresses are not used. For example, if pins EPI0S27 and EPI0S26 are used for device 1 and 0 respectively, then address 0x6800.0000 accesses device 0 (device 1 has its CE high), and 0x6400.0000 accesses device 1 (device 0 has its CE high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as $\overline{CE}$ because the address remains stable after the ALE strobe. The subsequent $\overline{WR}$ or $\overline{RD}$ signals write or read when ALE is low thus providing $\overline{CE}$ functionality.

Figure 17-33. EPI Host-Bus 16 Configuration (EPIHB16CFG) Register [offset 0x010]

31	30	29	28	27	26	25	24
CLKGATE	CLKGATEI	CLKINV	RDYEN	IRDYINV	Reserved		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0		
23	22	21	20	19	18	17	16
XFFEN	XFEEN	WRHIGH	RDHIGH	ALEHIGH	Reserved		BURST
R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R-0		R/W-0
15	14	13	12	11	10	9	8
MAXWAIT							
R/W-1							
7	6	5	4	3	2	1	0
WRWS		RDWS		Reserved	BSEL	MODE	
R/W-0		R/W-0		R-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-19. EPI Host-Bus 16 Configuration (EPIHB16CFG) Register Field Descriptions

Bit	Field	Value	Description
31	CLKGATE		Clock Gated
		0	The EPI clock is free running..
		1	The EPI clock is held low.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-19. EPI Host-Bus 16 Configuration (EPIHB16CFG) Register Field Descriptions (continued)

Bit	Field	Value	Description
30	CLKGATEI		Clock Gated when Idle
		0	The EPI clock is free running
			Note that EPI0S32 is an iRDY signal if RDYEN is set. CLKGATEI is ignored if CLKPIN is 0 or if the COUNT0 field in the EPIBAUD register is cleared.
		1	The EPI clock is output only when there is data to write or read (current transaction); otherwise the EPI clock is held low
29	CLKINV		Invert Output Clock Enable
		0	No effect
		1	Invert EPI clock to ensure the rising edge is centered for outbound signal's setup and hold. Inbound signal is captured on rising edge EPI clock.
28	RDYEN		Input Ready Enable
		0	No effect
		1	An external ready can be used to control the continuation of the current access. If this bit is set and the iRDY signal (EPIS032) is low, the current access is stalled.
27	IRDYINV		Input Ready Invert
		0	No effect
		1	Invert the polarity of incoming external ready (iRDY signal). If this bit is set and the iRDY signal (EPIS032) is high the current access is stalled.
26-24	Reserved		Reserved
23	XFFEN		External FIFO FULL Enable
		0	No effect.
		1	An external FIFO full signal can be used to control write cycles. If this bit is set and the FFULL full signal is high, XFIFO writes are stalled.
22	XFEEN		External FIFO EMPTY Enable
		0	No effect
		1	An external FIFO empty signal can be used to control read cycles. If this bit is set and the FEMPTY signal is high, XFIFO reads are stalled.
21	WRHIGH		WRITE Strobe Polarity
		0	The WRITE strobe for $\overline{CS0}$ is $\overline{WR}$ (active Low).
		1	The WRITE strobe for $\overline{CS0}$ is WR (active High).
20	RDHIGH		READ Strobe Polarity
		0	The READ strobe for $\overline{CS0}$ is $\overline{RD}$ (active Low).
		1	The READ strobe for $\overline{CS0}$ is RD (active High).
19	ALEHIGH		ALE Strobe Polarity
		0	The address latch strobe for $\overline{CS0}$ accesses is $\overline{ALE}$ (active Low).
		1	The address latch strobe for $\overline{CS0}$ accesses is ALE (active High).
18-17	Reserved		Reserved
16	BURST		Burst Mode
			Burst mode must be used with an ALE-enabled interface. Burst mode must be used with ADMUX, which is configured by the MODE field in the EPIHB16CFG register. Note: Burst mode is optimized for word-length accesses.
		0	Burst mode is disabled.
		1	Burst mode is enabled for $\overline{CS0}$ or single chip access.
15-8	MAXWAIT		Maximum Wait This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY). When the MAXWAIT value is reached the ERRRIS interrupt status bit is set in the EPIRIS register. When this field is clear, the transaction can be held off forever without a system interrupt. **Note:** When the MODE field is configured to be 0x2 and the BLKEN bit is set in the EPICFG register, enabling HB8 mode, this field defaults to 0xFF.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-19. EPI Host-Bus 16 Configuration (EPIHB16CFG) Register Field Descriptions (continued)

Bit	Field	Value	Description
7-6	WRWS		Write Wait States This field adds wait states to the data phase of $\overline{CS0}$ (the address phase is not affected). The effect is to delay the rising edge of $\overline{WR}$ (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit EPIHB16TIME register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is not applicable in BURST mode
		0x0	Active $\overline{WR}$ is 2 EPI clocks
		0x1	Active $\overline{WR}$ is 4 EPI clocks
		0x2	Active $\overline{WR}$ is 6 EPI clocks
		0x3	Active $\overline{WR}$ is 8 EPI clocks
			This field is used in conjunction with the EPIBAUD register.
5-4	RDWS		Read Wait States This field adds wait states to the data phase of $\overline{CS0}$ (the address phase is not affected). The effect is to delay the rising edge of $\overline{RD/OE}$ (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the EPIHB16TIME register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is not applicable in BURST mode.
		0x0	Active $\overline{RD}$ is 2 EPI clocks.
		0x1	Active $\overline{RD}$ is 4 EPI clocks.
		0x2	Active $\overline{RD}$ is 6 EPI clocks
		0x3	Active $\overline{RD}$ is 8 EPI clocks
			This field is used in conjunction with the EPIBAUD register
3	Reserved		Reserved
2	BSEL		Byte Select Configuration
			This bit enables byte select operation.
		0	No byte selects. Data is read and written as 16 bits.
		1	Enable Byte Selects
			Two EPI signals function as byte select signals to allow 8-bit transfers. See Table 17-7 for details on which EPI signals are used.
1-0	MODE		Host Bus Sub-Mode This field determines which of four Host Bus 8 sub-modes to use. Sub-mode use is determined by the connected external peripheral. When used with multiple chip select option and the CSBAUD bit is set to 1 in the EPIHB8CFG2 register, this configuration is for $\overline{CS0}$. If the multiple chip select option is enabled and CSBAUD is clear, all chip-selects use the MODE encoding programmed in this register.
		0x0	ADMUX – AD[15:0] Data and Address are muxed.
		0x1	ADNONMUX – D[15:0] Data and address are separate.
		0x2	Continuous Read - D15:0] This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of $\overline{OE}$ strobing.
		0x3	XFIFO – D[15:0] This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE.

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.7 EPI General-Purpose Configuration (EPIGPCFG) Register, offset 0x010

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIGPCFG, the MODE field must be 0x0.

The RD2CYC bit must be set at all times in General-Purpose mode to ensure proper operation.

The General-Purpose configuration register is used to configure the control, data, and address pins. This mode can be used for custom interfaces with FPGAs, CPLDs, and for digital data acquisition and actuator control. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the General-purpose mode is selected again, the register the values must be reinitialized.

This mode is designed for three general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs, with three sizes of data and optional addresses. Framing and clock-enable permit more optimized interfaces.
- General parallel GPIO. From 1 to 32 pins may be written or read, with the speed precisely controlled by the baud rate in the EPIBAUD register (when used with the NBRFIFO and/or the WFIFO) or by rate of accesses from software or µDMA.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock(free running or gated), a framing signal (with frame size),a ready input (to stretch transactions), read and write strobes, address of varying sizes, and data of varying sizes. Additionally, provisions are made for splitting address and data phases on the external interface.

Figure 17-34. EPI General-Purpose Configuration (EPIGPCFG) Register [offset 0x010]

31	30	29	28	27	26	25	24
CLKPIN	CLKGATE	Reserved	RDYEN	Reserved	FRM50	FRMCNT	
R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0x0	

23	22	21	20	19	18	17	16
FRMCNT		Reserved		WR2CYC	RD2CYC	Reserved	
R/W-0x0		R-0		R/W-0	R/W-0	R-0x0	

15	8	7	6	5	4	3	2	1	0
MAXWAIT		Reserved		ASIZE		Reserved		DSIZE	
R/W-0x00		R-0x0		R/W-0x0		R-0x0		R/W-0x0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-20. EPI General-Purpose Configuration (EPIGPCFG) Register Field Descriptions

Bit	Field	Value	Description
31	CLKPIN		Clock Pin The EPI clock is generated from the COUNT0 field in the EPIBAUD register (as is the system clock which is divided down from it). Note: If general purpose mode is configured to 32-bit data transfers (DSIZE = 0x3 in the EPICFG register), CLKPIN must be set to 0 to prevent clock-gated reads.
		0	No clock output.
		1	EPI0S31 functions as the EPI clock output.
30	CLKGATE		Clock Gated Note that EPI0S27 is an iRDY signal if RDYEN is set. CLKGATE is ignored if CLKPIN is 0 or if the COUNT0 field in the EPIBAUD register is cleared.
		0	The EPI clock is free running.
		1	The EPI clock is output only when there is data to write or read (current transaction); otherwise the EPI clock is held low.
29	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-20. EPI General-Purpose Configuration (EPIGPCFG) Register Field Descriptions (continued)

Bit	Field	Value	Description
28	RDYEN		Ready Enable The ready enable signal may only be used with a free-running EPI clock(CLKGATE=0).The external iRDY signal is sampled on the falling edge of the EPI clock. Setup and hold times must be met to ensure registration on the next falling EPI clock edge. This bit is ignored if CLKPIN is 0 or CLKGATE is 1.
		0	The external peripheral does not drive an iRDY signal and is assumed to be ready always.
		1	The external peripheral drives an iRDY signal into pin EPI0S27.
27	Reserved		Reserved
26	FRM50		50/50 Frame
		0	The FRAME signal is output as a single pulse, and then held low for the count.
		1	The FRAME signal is output as 50/50 duty cycle using count (see FRMCNT).
25-22	FRMCNT		Frame Count This field specifies the size of the frame in EPI clocks. The frame counter is used to determine the frame size. The count is FRMCNT+1.So, a FRMCNT of 0 forms a pure transaction valid signal (held high during transactions, low otherwise). A FRMCNT of 0 with FRM50 set inverts the FRAME signal on each transaction. A FRMCNT of 1 means the FRAME signal is inverted every other transaction; a value of 15 means every sixteenth transaction. If FRM50 is set, the frame is held high for FRMCNT+1 transactions, then held low for that many transactions, and so on. If FRM50 is clear, the frame is pulsed high for one EPI clock and then low for FRMCNT EPI clocks. This field is ignored if FRMPIN is 0.
21-20	Reserved		Reserved
19	WR2CYC		2-Cycle Writes When this bit is set, then the RW bit is forced to be set.
		0	Data is output on the same EPI clock cycle as the address.
		1	Writes are two EPI clock cycles long, with address on one EPI clock cycle (with the WR strobe asserted) and data written on the following EPI clock cycle (with WR strobe de-asserted). The next address (if any) is in the cycle following.
18	RD2CYC		2-Cycle Reads When this bit is set, then the RW bit is forced to be set. **Caution: This bit must be set at all times in General-Purpose mode to ensure proper operation.**
		0	Data is captured on the EPI clock cycle with READ strobe asserted.
		1	Reads are two EPI clock cycles, with address on one EPI clock cycle (with the RD strobe asserted) and data captured on the following EPI clock cycle (with the RD strobe de-asserted). The next address (if any) is in the cycle following.
17-16	Reserved		Reserved
15-8	MAXWAIT		Maximum Wait This field defines the maximum number of EPI clocks to wait while the iRDY signal (see RDYEN) is holding off a transaction. If this field is 0, the transaction is held forever. If the maximum wait of 255 clocks (MAXWAIT=0xFF) is exceeded, an error interrupt occurs and the transaction is aborted/ignored. **Note:** When the MODE field is configured to be 0x0 and the BLKEN bit is set in the EPICFG register, enabling General-Purpose mode, this field defaults to 0xFF.
7-6	Reserved		Reserved
5-4	ASIZE		Address Bus Size This field defines the size of the address bus. The address can be up to 4-bits wide with a 24-bit data bus, up to 12-bits wide with a 16-bit data bus, and up to 20-bits wide with an 8-bit data bus. If the full address bus is not used, use the least significant address bits. Any unused address bits can be used as GPIOs by clearing the AFSEL bit for the corresponding GPIOs. Also, if RDYEN is 1, then the address sizes are 1 smaller (3, 11, 19). The values are:
		0x0	No address.
		0x1	Up to 4 bits wide.
		0x2	Up to 12 bits wide. This size cannot be used with 24-bit data.
		0x3	Up to 20 bits wide. This size cannot be used with data sizes other than 8.
3-2	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-20. EPI General-Purpose Configuration (EPIGPCFG) Register Field Descriptions (continued)

Bit	Field	Value	Description
1-0	DSIZE		Size of Data Bus This field defines the size of the data bus (starting at EPI0S0). Subsets of these numbers can be created by clearing the AFSEL bit for the corresponding GPIOs. Note that size 32 may not be used with clock, frame, address, or other control. The values are:
		0x0	8 Bits Wide (EPI0S0 to EPI0S7)
		0x1	16 Bits Wide (EPI0S0 to EPI0S15)
		0x2	24 Bits Wide (EPI0S0 to EPI0S23)
		0x3	32 Bits Wide (EPI0S0 to EPI0S31) This size may not be used with an EPI clock. This value is normally used for acquisition input and actuator control as well as other general-purpose uses that require 32 bits per direction.

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.8 EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2) Register, Offset 0x014

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIHB8CFG2, the MODE field must be 0x2.

This register is used to configure operation while in Host-Bus 8 mode. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the Host-Bus 8 mode is selected again, the values must be reinitialized.

Figure 17-35. EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2) Register [offset 0x014]

31	30	29	28	27	26	25	24
WORD	Reserved			CSCFGEXT	CSBAUD	CSCFG	
R/W-0	R-0			R/W-0	R/W-0	R/W-0	

23	22	21	20	19	18	17	16
Reserved		WRHIGH	RDHIGH	ALEHIGH	Reserved		
R-0		R/W-0	R/W-0	R/W-1	R-0		

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
WRWS		RDWS		Reserved		MODE	
R/W-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17-21. EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2) Register Field Descriptions

Bit	Field	Value	Description
31	WORD		Word Access Mode By default, the EPI controller uses data bits [7:0] for Host-Bus 8 accesses. When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:8]. When WORD is set, short and long variables can be used in C programs.
		0	Word Access mode is disabled
		1	Word Access mode is enabled
30-28	Reserved		Reserved
27	CSCFGEXT		Chip Select Extended Configuration This field is used in conjunction with CSCFG, to extend the chip select options, and ALE format. The values 0x0 through 0x3 are from the CSCFG field. The CSCFGEXT bit extends the values to 0x7.
		0	CSCFG bit field is used in chip select configuration
		1	The CSCFG bit field is extended with CSCFGEXT representing the MSB.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-21. EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2) Register Field Descriptions (continued)

Bit	Field	Value	Description
26	CSBAUD		Chip Select Baud Rate and Multiple Sub-Mode Configuration enable This bit is only valid when the CSCFGEXT + CSCFG field is programmed to 0x2 or 0x3, 0x5 or 0x6. This bit configures the baud rate settings for $\overline{CS0}$, $\overline{CS1}$, $\overline{CS2}$, and $\overline{CS3}$. This bit must also be set to allow different sub-mode configurations on chip-selects. If this bit is clear, all chip-select sub-modes are based on the MODE encoding defined in the EPI8HBCFG register. If the CSBAUD bit is set in the EPIHBnCFG2 register and dual- or quad-chip selects are enabled, then the individual chip selects can use different clock frequencies, wait states and strobe polarity .
		0	Same Baud Rate and Same Sub-Mode All $\overline{CS}$ use the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD register and the sub-mode programmed in the MODE field of the EPIHB8CFG register.
		1	Different Baud Rates $\overline{CS0}$ uses the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD register. $\overline{CS1}$ uses the baud rate defined by the COUNT1 field in the EPIBAUD register. $\overline{CS2}$ uses the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD2 register. $\overline{CS3}$ uses the baud rate defined by the COUNT1 field in the EPIBAUD2 register. In addition, the sub-modes for each chip select are individually programmed in their respective EPIHB8CFGn registers.
25-24	CSCFG		Chip Select Configuration This field controls the chip select options, including an ALE format, a single chip select, two chip selects, and an ALE combined with two chip selects. These bits are also used in combination with the CSCFGEXT bit for further configurations, including quad- chip select.
		0x0	ALE Configuration EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (HB8MODE field in the EPIHB8CFG register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle.
		0x1	$\overline{CS}$ Configuration EPI0S30 is used as a Chip Select ($\overline{CS}$). When using this mode, the address and data are generally not muxed (HB8MODE field in the EPIHB8CFG register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when $\overline{CS}$ is low.
		0x2	Dual $\overline{CS}$ Configuration EPI0S30 is used as $\overline{CS0}$ and EPI0S27 is used as $\overline{CS1}$. Whether $\overline{CS0}$ or $\overline{CS1}$ is asserted is determined by two methods. If only external RAM or external PER is enabled in the address map, the most significant address bit for a respective external address map controls $\overline{CS0}$ or $\overline{CS1}$. If both external RAM and external PER is enabled, $\overline{CS0}$ is mapped to PER and $\overline{CS1}$ is mapped to RAM. This configuration can be used for a RAM bank split between two devices as well as when using both an external RAM and an external peripheral.
		0x3	ALE with Dual $\overline{CS}$ Configuration EPI0S30 is used as address latch (ALE), EPI0S27 is used as $\overline{CS1}$, and EPI0S26 is used as $\overline{CS0}$. Whether $\overline{CS0}$ or $\overline{CS1}$ is asserted is determined by the most significant address bit for a respective external address map.
23-22	Reserved		Reserved
21	WRHIGH		$\overline{CS1}$ WRITE Strobe Polarity This field is used if the CSBAUD bit in the EPIHB8CFG2 register is enabled.
		0	The WRITE strobe for $\overline{CS1}$ accesses is $\overline{WR}$ (active Low).
		1	The WRITE strobe for $\overline{CS1}$ accesses is WR (active High).
20	RDHIGH		$\overline{CS1}$ READ Strobe Polarity This field is used if the CSBAUD bit in the EPIHB8CFG2 register is enabled.
		0	The READ strobe for $\overline{CS1}$ accesses is $\overline{RD}$ (active Low).
		1	The READ strobe for $\overline{CS1}$ accesses is RD (active High).
19	ALEHIGH		$\overline{CS1}$ ALE Strobe Polarity This field is used if the CSBAUD bit in the EPIHB8CFG2 register is enabled.
		0	The address latch strobe for $\overline{CS1}$ accesses is $\overline{ALE}$ (active Low).
		1	The address latch strobe for $\overline{CS1}$ accesses is ALE (active High).
18-8	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-21. EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2) Register Field Descriptions (continued)

Bit	Field	Value	Description
7-6	WRWS		$\overline{CS1}$ Write Wait States This field adds wait states to the data phase of $\overline{CS1}$ accesses (the address phase is not affected). The effect is to delay the rising edge of $\overline{WR}$ (or the falling edge of WR). Each wait state encoding adds 2 EPI clock cycles to the access time. The WRWSM bit in the EPIHB8TIME2 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register. This field is used in conjunction with the EPIBAUD register and is not applicable in BURST mode.
		0x0	Active $\overline{WR}$ is 2 EPI clocks
		0x1	Active $\overline{WR}$ is 4 EPI clocks
		0x2	Active $\overline{WR}$ is 6 EPI clocks
		0x3	Active $\overline{WR}$ is 8 EPI clocks
5-4	RDWS		$\overline{CS1}$ Read Wait States This field adds wait states to the data phase of $\overline{CS1}$ accesses (the address phase is not affected). The effect is to delay the rising edge of $\overline{RD}$/$\overline{OE}$ (or the falling edge of RD). Each wait state encoding adds 2 EPI clock cycles to the access time. The RDWSM bit in the EPIHB8TIME2 register can decrease the number of states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register. This field is used in conjunction with the EPIBAUD register and is not applicable in BURST mode.
		0x0	Active $\overline{RD}$ is 2 EPI clocks
		0x1	Active $\overline{RD}$ is 4EPI clocks
		0x2	Active $\overline{RD}$ is 4 EPI clocks
		0x3	Active $\overline{RD}$ is 8 EPI clocks
3-2	Reserved		Reserved
1-0	MODE		$\overline{CS1}$ Host Bus Sub-Mode This field determines which Host Bus 8 sub-mode to use for $\overline{CS1}$. Sub-mode use is determined by the externally connected peripheral or memory. **Note:** The CSBAUD bit must be set to enable this $\overline{CS1}$ MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the EPIHB8CFG register.
		0x0	ADMUX – AD[7:0] Data and Address are muxed.
		0x1	ADNONMUX – D[7:0] Data and address are separate.
		0x2	Continuous Read - D[7:0] This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of $\overline{OE}$ strobing.
		0x3	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.9 EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2) Register, offset 0x014

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIHB16CFG2, the MODE field must be 0x3.

This register is used to configure operation while in Host-Bus 16 mode. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the Host-Bus 16 mode is selected again, the values must be reinitialized.

Figure 17-36. EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2) Register [offset 0x014]

31	30	29	28	27	26	25	24
WORD	Reserved			CSCFGEXT	CSBAUD	CSCFG	
R/W-0	R-0			R/W-0	R/W-0	R/W-0	

23	22	21	20	19	18	17	16
Reserved		WRHIGH	RDHIGH	ALEHIGH	Reserved		BURST
R-0		R/W-0	R/W-0	R/W-1	R-0		R/W-0

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
WRWS		RDWS		Reserved		MODE	
R/W-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17-22. EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2) Register Field Descriptions

Bit	Field	Value	Description
31	WORD		Word Access Mode By default, the EPI controller uses data bits [15:0] for Host-Bus 16 accesses. When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:16]. When WORD is set, long variables can be used in C programs.
		0	Word Access mode is disabled.
		1	Word Access mode is enabled.
30-28	Reserved		Reserved
27	CSCFGEXT		Chip Select Extended Configuration This field is used in conjunction with CSCFG, to extend the chip select options, and ALE format. The values 0x0 through 0x3 are from the CSCFG field. The CSCFGEXT bit extends the values to 0x7.
		0	CSCFG bit field is used in chip select configuration.
		1	The CSCFG bit field is extended with CSCFGEXT representing the MSB.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-22. EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2) Register Field Descriptions (continued)

Bit	Field	Value	Description
26	CSBAUD		Chip Select Baud Rate and Multiple Sub-Mode Configuration enable This bit is only valid when the CSCFGEXT + CSCFG field is programmed to 0x2 or 0x3, 0x5 or 0x6. This bit configures the baud rate settings for $\overline{CS0}$, $\overline{CS1}$, $\overline{CS2}$, and $\overline{CS3}$. This bit must also be set to allow different sub-mode configurations on chip-selects. If this bit is clear, all chip-select sub-modes are based on the MODE encoding defined in the EPI8HBCFG register. If the CSBAUD bit is set in the EPIHBnCFG2 register and dual- or quad-chip selects are enabled, then the individual chip selects can use different clock frequencies, wait states and strobe polarity.
		0	Same Baud Rate and Same Sub-Mode All $\overline{CS}$ use the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD register and the sub-mode programmed in the MODE field of the EPIHB16CFG register.
		1	Different Baud Rates $\overline{CS0}$ uses the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD register. $\overline{CS1}$ uses the baud rate defined by the COUNT1 field in the EPIBAUD register. $\overline{CS2}$ uses the baud rate for the external bus that is defined by the COUNT0 field in the EPIBAUD2 register. $\overline{CS3}$ uses the baud rate defined by the COUNT1 field in the EPIBAUD2 register. In addition, the sub-modes for each chip select are individually programmed in their respective EPIHB16CFGn registers.
25-24	CSCFG		Chip Select Configuration This field controls the chip select options, including an ALE format, a single chip select, two chip selects, and an ALE combined with two chip selects. These bits are also used in combination with the CSCFGEXT bit for further configurations, including quad- chip select.
		0x0	ALE Configuration EPI0S30 is used as an address latch (ALE). When using this mode, the address and data should be muxed (HB16MODE field in the EPIHB16CFG register should be configured to 0x0). If needed, the address can be latched by external logic.
		0x1	$\overline{CS}$ Configuration EPI0S30 is used as a Chip Select ($\overline{CS}$). When using this mode, the address and data should not be muxed (MODE field in the EPIHB16CFG register should be configured to 0x1). In this mode, the WR signal (EPI0S29) and the RD signal (EPI0S28) are used to latch the address when $\overline{CS}$ is low.
		0x2	Dual $\overline{CS}$ Configuration EPI0S30 is used as $\overline{CS0}$ and EPI0S27 is used as $\overline{CS1}$. Whether $\overline{CS0}$ or $\overline{CS1}$ is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral.
		0x3	ALE with Dual $\overline{CS}$ Configuration EPI0S30 is used as address latch (ALE), EPI0S27 is used as $\overline{CS1}$, and EPI0S26 is used as $\overline{CS0}$. Whether $\overline{CS0}$ or $\overline{CS1}$ is asserted is determined by the most significant address bit for a respective external address map.
23-22	Reserved		Reserved
21	WRHIGH		$\overline{CS1}$ WRITE Strobe Polarity This field is used if CSBAUD bit of the EPIHB16CFG2 register is enabled.
		0	The WRITE strobe for $\overline{CS1}$ accesses is $\overline{WR}$ (active Low).
		1	The WRITE strobe for $\overline{CS1}$ accesses is WR (active High).
20	RDHIGH		$\overline{CS1}$ READ Strobe Polarity This field is used if CSBAUD bit of the EPIHB16CFG2 register is enabled
		0	The READ strobe for $\overline{CS1}$ accesses is $\overline{RD}$ (active Low).
		1	The READ strobe for $\overline{CS1}$ accesses is RD (active High).
19	ALEHIGH		$\overline{CS1}$ ALE Strobe Polarity This field is used if CSBAUD bit of the EPIHB16CFG2 register is enabled
		0	The address latch strobe for $\overline{CS1}$ accesses is $\overline{ALE}$ (active Low).
		1	The address latch strobe for $\overline{CS1}$ accesses is ALE (active High).
18-17	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-22. EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2) Register Field Descriptions (continued)

Bit	Field	Value	Description
16	BURST		$\overline{CS1}$ Burst Mode Burst mode must be used with an ALE which is configured by programming the CSCFG and CSCFGEXT fields in the EPIHB16CFG2 register. Burst mode must be used in ADMUX, which is set by the MODE field in EPIHB16CFG2. Note: Burst mode is optimized for word-length accesses.
		0	Burst mode is disabled.
		1	Burst mode is enabled for $\overline{CS1}$.
15-8	Reserved		Reserved
7-6	WRWS		$\overline{CS1}$ Write Wait States This field adds wait states to the data phase of $\overline{CS1}$ accesses (the address phase is not affected). The effect is to delay the rising edge of $\overline{WR}$ (or the falling edge of WR). Each wait state encoding adds 2 EPI clock cycles to the access time. The WRWSM bit in the EPIHB16TIME2 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB16CFG2 register. This field is used in conjunction with the EPIBAUD register and is not applicable in BURST mode.
		0x0	Active $\overline{WR}$ is 2 EPI clocks
		0x1	Active $\overline{WR}$ is 4 EPI clocks
		0x2	Active $\overline{WR}$ is 6 EPI clocks
		0x3	Active $\overline{WR}$ is 8 EPI clocks
5-4	RDWS		$\overline{CS1}$ Read Wait States This field adds wait states to the data phase of $\overline{CS1}$ accesses (the address phase is not affected). The effect is to delay the rising edge of $\overline{RD}$/$\overline{OE}$ (or the falling edge of RD). Each wait state encoding adds 2 EPI clock cycles to the access time. The RDWSM bit in the EPIHB16TIME2 register can decrease the number of states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB16CFG2 register. This field is used in conjunction with the EPIBAUD register and is not applicable in BURST mode.
		0x0	Active $\overline{RD}$ is 2 EPI clocks
		0x1	Active $\overline{RD}$ is 4 EPI clocks
		0x2	Active $\overline{RD}$ is 6 EPI clocks
		0x3	Active $\overline{RD}$ is 8 EPI clocks
3-2	Reserved		Reserved
1-0	MODE		$\overline{CS1}$ Host Bus Sub-Mode This field determines which Host Bus 16 sub-mode to use for $\overline{CS1}$. Sub-mode use is determined by the connected external peripheral. See Table 17-8 for information on how this bit field affects the operation of the EPI signals. When used with multiple chip select option this configuration is for $\overline{CS1}$. **Note:** The CSBAUD bit must be set to enable this $\overline{CS1}$ MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the EPIHB16CFG register.
		0x0	ADMUX – AD[15:0] Data and Address are muxed
		0x1	ADNONMUX – D[15:0] Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.
		0x2	Continuous Read - D[15:0] This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of $\overline{OE}$ strobing. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available.
		0x3	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.10 EPI General-Purpose Configuration 2 (EPIGPCFG2) Register, offset 0x014

NOTE: The MODE field in the EPICFG register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access EPIGPCFG2, the MODE field must be 0x0.

This register is used to configure operation while in General-Purpose mode. Note that this register is reset when the MODE field in the EPICFG register is changed. If another mode is selected and the General-Purpose mode is selected again, the values must be reinitialized.

Figure 17-37. EPI General-Purpose Configuration 2 (EPIGPCFG2)Register [offset 0x014]

31	30	0
WORD	Reserved	
R/W-0	R-0	

LEGENR/W-0D: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-23. EPI General-Purpose Configuration 2 (EPIGPCFG2) Register Field Descriptions

Bit	Field	Value	Description
31	WORD		Word Access Mode By default, the EPI controller uses data bits [7:0] when the DSIZE field in the EPIGPCFG register is 0x0; data bits [15:0] when the DSIZE field is 0x1; data bits [23:0] when the DSIZE field is 0x2; and data bits [31:0] when the DSIZE field is 0x3. When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:8] for DSIZE=0x0 and bits [31:16] for DSIZE=0x1. For DSIZE=0x2 or 0x3, this bit must be clear.
		0	Word Access mode is disabled.
		1	Word Access mode is enabled.
30-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.11 EPI Address Map (EPIADDRMAP) Register, offset 0x01C

This register enables address mapping. The EPI controller can directly address memory and peripherals. In addition, the EPI controller supports address mapping to allow indirect accesses in the External RAM and External Peripheral areas.

If the external device is a peripheral, including a FIFO or a directly addressable device, the EPSZ and EPADR bit fields should be configured for the address space. If the external device is SDRAM, SRAM, or NOR Flash memory, the ERADR and ERSZ bit fields should be configured for the address space.

If one of the dual-chip select modes is selected (CSCFGEXT is 0x0 and CSCFG is 0x2 or 0x3 in the EPIHBnCFG2 register), both chip selects can share the peripheral or the memory space, or one chip select can use the peripheral space and the other can use the memory space. In the EPIADDRMAP register, if the EPADR field is not 0x0, the ECADR field is 0x0, and the ERADR field is 0x0, then the address specified by EPADR is used for both chip selects, with $\overline{CS0}$ being asserted when the MSB of the address range is 0 and $\overline{CS1}$ being asserted when the MSB of the address range is 1. If the ERADR field is not 0x0, the ECADR field is 0x0, and the EPADR field is 0x0, then the address specified by ERADR is used for both chip selects, with the MSB performing the same delineation. If both the EPADR and the ERADR are not 0x0 and the ECADR field is 0x0, then $\overline{CS0}$ is asserted for either address range defined by EPADR and $\overline{CS1}$ is asserted for either address range defined by ERADR. The two chip selects can also be shared between the code space and memory or peripheral space. If the ECADR field is 0x1, ERADR field is 0x0, and the EPADR field is not 0x0, then $\overline{CS0}$ is asserted for the address range defined by ECADR and $\overline{CS1}$ is asserted for either address range defined by EPADR. If the ECADR field is 0x1, EPADR field is 0x0, and the ERADR field is not 0x0, then $\overline{CS0}$ is asserted for the address range defined by ECADR and $\overline{CS1}$ is asserted for either address range defined by ERADR.

If one of the quad-chip-select modes is selected (CSCFGEXT is 0x1 and CSCFG is 0x2 or 0x3 in the EPIHBnCFG2 register), both the peripheral and the memory space must be enabled. In the EPIADDRMAP register, the EPADR field is 0x3, the ERADR field is 0x3, and the ECADR field is 0x0. In this case, $\overline{CS0}$ maps to 0x6000.0000; $\overline{CS1}$ maps to 0x8000.0000; $\overline{CS2}$ maps to 0xA000.0000; and $\overline{CS3}$ maps to 0xC000.0000. The MODE field of the EPIHBnCFGn registers configures the interface for the individual chip selects, which support ADMUX or ADNOMUX. If the CSBAUD bit is clear, all chip selects use the mode configured in the MODE bit field of the EPIHBnCFG register. Table 17-3 gives a detailed explanation of chip select address range mappings based on which combinations of peripheral and memory space are enabled.

Figure 17-38. EPI Address Map (EPIADDRMAP) Register [offset 0x01C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0x0000.00															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ECSZ		ECADR		EPSZ		EPADR		ERSZ		ERADR	
R-0				R/W-0		R/W-0		R/W-0x0		R/W-0x0		R/W-0x0		R/W-0x0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17-24. EPI Address Map (EPIADDRMAP) Register Field Descriptions

Bit	Field	Value	Description
31-12	Reserved		Reserved
11-10	ECSZ		External Code Size This field selects the size of the external code area. If the size of the external code area is larger, a bus fault occurs. If the size of the external code area is smaller, it wraps (upper address bits unused). **Note:** When not using byte selects in Host-Bus 16, data is accessed on 2-byte boundaries. As a result, the available address space is double the amount shown below.
		0x0	256 bytes; lower address range: 0x00 to 0xFF
		0x1	64KB; lower address range: 0x0000 to 0xFFFF
		0x2	16MB; lower address range: 0x00.0000 to 0xFF.FFFF
		0x3	256MB; lower address range: 0x000.0000 to 0xFFF.FFFF

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-24. EPI Address Map (EPIADDRMAP) Register Field Descriptions (continued)

Bit	Field	Value	Description
9-8	ECADR		External Code Address This field selects address mapping for the external code area.
		0x0	Not mapped
		0x1	At 0x1000.0000
		0x2	Reserved
		0x3	Reserved
7-6	EPSZ		External Peripheral Size This field selects the size of the external peripheral. If the size of the external peripheral is larger, a bus fault occurs. If the size of the external peripheral is smaller, it wraps (upper address bits unused). **Note:** When not using byte selects in Host-Bus 16, data is accessed on 2-byte boundaries. As a result, the available address space is double the amount shown below.
		0x0	256 bytes; lower address range: 0x00 to 0xFF
		0x1	64 KB; lower address range: 0x0000 to 0xFFFF
		0x2	16 MB; lower address range: 0x00.0000 to 0xFF.FFFF
		0x3	512 MB; lower address range: 0x000.0000 to 0x1FFF.FFFF
5-4	EPADR		External Peripheral Address This field selects address mapping for the external peripheral area.
		0x0	Not mapped.
		0x1	At 0xA000.0000
		0x2	At 0xC000.0000
		0x3	Only to be used with Host Bus quad chip select. In quad chip select mode, $\overline{CS2}$ maps to 0xA000.0000 and $\overline{CS3}$ maps to 0xC000.0000.
3-2	ERSZ		External RAM Size This field selects the size of mapped RAM. If the size of the external memory is larger, a bus fault occurs. If the size of the external memory is smaller, it wraps (upper address bits unused):
		0x0	256 bytes; lower address range: 0x00 to 0xFF
		0x1	64KB; lower address range: 0x0000 to 0xFFFF
		0x2	16MB; lower address range: 0x00.0000 to 0xFF.FFFF
		0x3	512MB; lower address range: 0x000.0000 to 0x1FFF.FFFF
1-0	ERADR		External RAM Address Selects address mapping for external RAM area.
		0x0	Not mapped.
		0x1	At 0x6000.0000
		0x2	At 0x8000.0000
		0x3	Only to be used with Host Bus quad chip select. In quad chip select mode, $\overline{CS0}$ maps to 0x6000.0000 and $\overline{CS1}$ maps to 0x8000.0000

17.11.12 EPI Read Size 0 (EPIRSIZE0) Register and EPI Read Size 1 (EPIRSIZE1) Register, offset 0x020 and 0x030

The SIZE field must match the external data width as configured in the EPIHBnCFG or EPIGPCFG register if the WORD bit is clear in the EPIHBnCFG2 or EPIGPCFG2 register. If the WORD bit is set, the SIZE field must be greater than or equal to the external data width.

SDRAM mode uses a 16-bit data interface. If SIZE is 0x1, data is returned on the least significant bits (D[7:0]), and the remaining bits D[31:8] are all zeros, therefore the data on bits D[15:8] is lost. If SIZE is 0x2, data is returned on the least significant bits (D[15:0]), and the remaining bits D[31:16] are all zeros.

Note that changing this register while a read is active has an unpredictable effect.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 17-39. EPI Read Size 0 (EPIRSIZE0) Register [offset 0x020] and EPI Read Size 1 (EPIRSIZE1) Register [offset 0x030]

31	2	1	0
Reserved		SIZE	
R-0x0000.000		R/W-0x3	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-25. EPI Read Size 0 (EPIRSIZE0) Register and EPI Read Size 1 (EPIRSIZE1) Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1-0	SIZE		Current Size
		0x0	Reserved
		0x1	Byte (8 bits)
		0x2	Half-word (16 bits)
		0x3	Word (32 bits)

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.13 EPI Read Address 0 (EPIRADDR0) Register and EPI Read Address 1 (EPIRADDR1) Register, offset 0x024 and 0x034

This register holds the current address value. When performing non-blocking reads via the EPIRPSTDn registers, this register's value forms the address (when used by the mode). That is, when an EPIRPSTDn register is written with a non-0 value, this register is used as the first address. After each read, it is incremented by the size specified by the corresponding EPIRSIZEn register. Thus at the end of a read, this register contains the next address for the next read. For example, if the last read was 0x20, and the size is word, then the register contains 0x24. When a non-blocking read is cancelled, this register contains the next address that would have been read had it not been cancelled. For example, if reading by bytes and 0x103 had been read but not 0x104, this register contains 0x104. In this manner, the system can determine the number of values in the NBRFIFO to drain..

Note that changing this register while a read is active has an unpredictable effect due to race condition.

Figure 17-40. EPI Read Address 0 (EPIRADDR0) Register [offset 0x024] and EPI Read Address 1 (EPIRADDR1) Register [offset 0x034]

31	0
ADDR	
R/W-0x0000.0000	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17-26. EPI Read Address 0 (EPIRADDR0) Register and EPI Read Address 1 (EPIRADDR1) Register Field Descriptions

Bit	Field	Value	Description
31-29	Reserved		Reserved
31-0	ADDR		Current Address Next address to read

17.11.14 EPI Non-Blocking Read Data 0 (EPIRPSTD0) Register and EPI Non-Blocking Read Data 1 (EPIRPSTD1) Register, offset 0x028 and 0x038

This register sets up a non-blocking read via the external interface. A non-blocking read is started by writing to this register with the count (other than 0). Clearing this register terminates an active non-blocking read as well as cancelling any that are pending. This register should always be cleared before writing a value other than 0; failure to do so can cause improper operation. Note that both NBR channels can be enabled at the same time, but NBR channel 0 has the highest priority and channel 1 does not start until channel 0 is finished.

The first address is based on the corresponding EPIRADDRn register. The address register is incremented by the size specified by the EPIRSIZEn register after each read. If the size is less than a word, only the least significant bits of data are filled into the NBRFIFO; the most significant bits are clearedhe first address is based on the corresponding EPIRADDRn register. The address register is incremented by the size specified by the EPIRSIZEn register after each read. If the size is less than a word, only the least significant bits of data are filled into the NBRFIFO; the most significant bits are cleared.

Note that all three registers may be written using one STM instruction, such as with a structure copy in C/C++.

The data may be read from the EPIREADFIFO register after the read cycle is completed. The interrupt mechanism is normally used to trigger the FIFO reads via ISR orµDMA.

If the countdown has not reached 0 and the NBRFIFO is full, the external interface waits until a NBRFIFO entry becomes available to continue.

Note: if a blocking read or write is performed through the address mapped area (at 0x6000.0000 through 0xDFFF.FFFF), any current non-blocking read is paused (at the next safe boundary), and the blocking request is inserted. After completion of any blocking reads or writes, the non-blocking reads continue from where they were paused.

Copyright © 2012–2019, Texas Instruments Incorporated

The other way to read data is via the address mapped locations (see the EPIADDRMAP register), but this method is blocking (core or µDMA waits until result is returned).

To cancel a non-blocking read, clear this register. To make sure that all values read are drained from the NBRFIFO, the EPISTAT register must be consulted to be certain that bits NBRBUSY and ACTIVE are cleared. One of these registers should not be cleared until either the other EPIRPSTDn register becomes active or the external interface is not busy. At that point, the corresponding EPIRADDRn register indicates how many values were read.

Figure 17-41. EPI Non-Blocking Read Data 0 (EPIRPSTD0) Register [offset 0x028] and EPI Non-Blocking Read Data 1 (EPIRPSTD1) Register [offset 0x038]

31	13	12	0
Reserved		POSTCNT	
R-0x0000.0		R/W-0x000	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-27. EPI Non-Blocking Read Data 0 (EPIRPSTD0) Register and EPI Non-Blocking Read Data 1 (EPIRPSTD1) Register Field Descriptions

Bit	Field	Value	Description
31-13	Reserved		Reserved
12-0	POSTCNT		Post Count A write of a non-zero value starts a read operation for that count. Note that it is the software's responsibility to handle address wraparound. Reading this register provides the current count. A write of 0 cancels a non-blocking read (whether active now or pending).Prior to writing a non-zero value, this register must first be cleared.

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.15 EPI Status (EPISTAT) Register, offset 0x060

This register indicates which non-blocking read register is currently active; it also indicates whether the external interface is busy performing a write or non-blocking read (it cannot be performing a blocking read, as the bus would be blocked and as a result, this register could not be accessed).

This register is useful to determine which non-blocking read register is active when both are loaded with values and when implementing sequencing or sharing.

This register is also useful when canceling non-blocking reads, as it shows how many values were read by the canceled side.

Figure 17-42. EPI Status (EPISTAT) Register [offset 0x060]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0x0000.00															

15	14	13	12	11	10	9	8
Reserved						CELOW	XFFULL
R-0x0000.00						R-0	R-0

7	6	5	4	3	2	1	0
XFEMPTY	INITSEQ	WBUSY	NBRBUSY	Reserved			ACTIVE
R-0	R-0	R-0	R-0	R-0x0			R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-28. EPI Status (EPISTAT) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Reserved
9	CELOW		Clock Enable Low This bit provides information on the clock status when in general-purpose mode and the RDYEN bit is set.
		0	The external device is not gating the clock.
		1	The external device is gating the clock (iRDY is low). Attempts to read or write in this situation are stalled until the clock is enabled or the counter times out as specified by the MAXWAIT field.
8	XFFULL		External FIFO Full This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus n mode with the XFFEN bit set in the EPIHBnCFG register. The EPI0S26 signal reflects the status of this bit.
		0	The external device is not gating the clock.
		1	The XFIFO is signaling as full (the FIFO full signal is high). Attempts to write in this case are stalled until the XFIFO full signal goes low or the counter times out as specified by the MAXWAIT field.
7	XFEMPTY		External FIFO Empty This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus n mode with the XFEEN bit set in the EPIHBnCFG register. The EPI0S27 signal reflects the status of this bit.
		0	The external device is not gating the clock.
		1	The XFIFO is signaling as empty (the FIFO empty signal is high). Attempts to read in this case are stalled until the XFIFO empty signal goes low or the counter times out as specified by the MAXWAIT field.
6	INITSEQ		Initialization Sequence
		0	The SDRAM interface is not in the wakeup period.
		1	The SDRAM interface is running through the wakeup period (greater than 100 μs). If an attempt is made to read or write the SDRAM during this period, the access is held off until the wakeup period is complete.
5	WBUSY		Write Busy
		0	The external interface is not performing a write.
		1	The external interface is performing a write.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-28. EPI Status (EPISTAT) Register Field Descriptions (continued)

Bit	Field	Value	Description
4	NBRBUSY		Non-Blocking Read Busy
		0	The external interface is not performing a non-blocking read.
		1	The external interface is performing a non-blocking read, or if the non-blocking read is paused due to a write.
3-1	Reserved		Reserved
0	ACTIVE		Register Active .
		0	If NBRBUSY is set, the EPIRPSTD0 register is active. If the NBRBUSY bit is clear, then neither EPIRPSTDx register is active.
		1	The EPIRPSTD1 register is active

17.11.16 EPI Read FIFO Count (EPIRFIFOCNT) Register, offset 0x06C

This register returns the number of values in the NBRFIFO (the data in the NBRFIFO can be read via the EPIREADFIFO register). A race is possible, but that only means that more values may come in after this register has been read.

Figure 17-43. EPI Read FIFO Count (EPIRFIFOCNT) Register [offset 0x06C]

31	4	3	0
Reserved		COUNT	
R-0x0000.000		R	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17-29. EPI Read FIFO Count (EPIRFIFOCNT) Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reserved
3-0	COUNT		FIFO Count Number of filled entries in the NBRFIFO.

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.17 EPI Read FIFO (EPIREADFIFO) Register and EPI Read FIFO Alias 1-7 (EPIREADFIFO1-7) Registers, offset 0x070 and 0x08C

This register returns the contents of the NBRFIFO or 0 if the NBRFIFO is empty. Each read returns the data that is at the top of the NBRFIFO, and then empties that value from the NBRFIFO. The alias registers can be used with the LDMIA instruction for more efficient operation (for up to 8 registers). See *Cortex™-M3 Instruction Set Technical Reference Manual* for more information on the LDMIA instruction.

Figure 17-44. EPI Read FIFO (EPIREADFIFO) Register [offset 0x070] and EPI Read FIFO Alias 1-7 (EPIREADFIFO1-7) Registers [offset 0x074 - 0x08C]

31	0
DATA	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-30. EPI Read FIFO (EPIREADFIFO) Register and EPI Read FIFO Alias 1-7 (EPIREADFIFO1-7) Registers Field Descriptions

Bit	Field	Value	Description
31-0	DATA		Reads Data This field contains the data that is at the top of the NBRFIFO. After being read, the NBRFIFO entry is removed.

17.11.18 EPI FIFO Level Selects (EPIFIFOLVL) Register, 0x200

This register allows selection of the FIFO levels which trigger an interrupt to the interrupt controller or, more efficiently, a DMA request to the µDMA. The NBRFIFO select triggers on fullness such that it triggers on match or above (more full) in order for the processor or the µDMA to extract the read data. The WFIFO triggers on emptiness such that it triggers on match or below (less entries) in order for the processor or the µDMA to insert more write data).

It should be noted that the FIFO triggers are not identical to other such FIFOs in these peripherals. In particular, empty and full triggers are provided to avoid wait states when using blocking operations.

The settings in this register are only meaningful if the µDMA is active or the interrupt is enabled.

Additionally, this register allows protection against writes stalling and notification of performing blocking reads which stall for extra time due to preceding writes. The two functions behave in a non-orthogonal way because read and write are not orthogonal.

The write error bit configures the system such that an attempted write to an already full WFIFO abandons the write and signals an error interrupt to prevent accidental latencies due to stalling writes.

The read error bit configures the system such that after a read has been stalled due to any preceding writes in the WFIFO, the error interrupt is generated. Note that the excess stall is not prevented, but an interrupt is generated after the fact to notify that it has happened.

Figure 17-45. EPI FIFO Level Selects (EPIFIFOLVL) Register [offset 0x200]

31	18	17	16
Reserved		WFERR	RSERR
R-0x000		R/W-0	R/W-0

15	7	6	4	3	2	0
Reserved		WRFIFO		Reserved	RDFIFO	
R-0x00		R/W-0x3		R-0	R/W-0x3	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-31. EPI FIFO Level Selects (EPIFIFOLVL) Register Field Descriptions

Bit	Field	Value	Description
31-18	Reserved		Reserved
17	WFERR		Write Full Error
		0	The Write Full error interrupt is disabled. Writes are stalled when the WFIFO is full until a space becomes available but an error is not generated. Note that the Cortex-M3 write buffer may hide that stall if no other memory transactions are attempted during that time.
		1	This bit enables the Write Full error interrupt (WTFULL in the EPIEISC register) to be generated when a write is attempted and the WFIFO is full. The write stalls until a WFIFO entry becomes available.
16	RSERR		Read Stall Error Note that the configuration of this bit has no effect on non-blocking reads.
		0	The Read Stalled error interrupt is disabled. Reads behave as normal and are stalled until any preceding writes have completed and the read has returned a result.
		1	This bit enables the Read Stalled error interrupt (RSTALL in the EPIEISC register) to be generated when a read is attempted and the WFIFO is not empty. The read is still stalled during the time the WFIFO drains, but this error notifies the application that this excess delay has occurred.
15-7	Reserved		Reserved
6-4	WRFIFO		Write FIFO This field configures the trigger point for the WFIFO.
		0x0	Interrupt is triggered while WRFIFO is empty. It will be deasserted when not empty. This encoding is optimized for burst of 4 writes
		0x1	Reserved
		0x2	Interrupt is triggered until there are only two slots available. Thus, trigger is deasserted when there are two WRFIFO entries present. This configuration is optimized for bursts of 2 .
		0x3	Interrupt is triggered until there is one WRFIFO entry available. This configuration expects only single writes.
		0x4	Trigger interrupt when WRFIFO is not full, meaning trigger will continue to assert until there are four entries in the WRFIFO.
		0x5-0x7	Reserved
3	Reserved		Reserved
2-0	RDFIFO		Read FIFO This field configures the trigger point for the NBRFIFO.
		0x0	Reserved
		0x1	Trigger when there are 1 or more entries in the NBRFIFO.
		0x2	Trigger when there are 2 or more entries in the NBRFIFO.
		0x3	Trigger when there are 4 or more entries in the NBRFIFO.
		0x4	Trigger when there are 6 or more entries in the NBRFIFO.
		0x5	Trigger when there are 7 or more entries in the NBRFIFO.
		0x6	Trigger when there are 8 entries in the NBRFIFO.
		0x7	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.19 EPI Write FIFO Count (EPIWFIFOCNT) Register, offset 0x204

This register contains the number of slots currently available in the WFIFO. This register may be used for polled writes to avoid stalling and for blocking reads to avoid excess stalling (due to undrained writes). An example use for writes may be:

```
for (idx = 0; idx < cnt; idx++) {
while (EPIWFIFOCNT== 0) ;
*ext_ram = *mydata++;
}
```

The above code ensures that writes to the address mapped location do not occur unless the WFIFO has room. Although polling makes the code wait (spinning in the loop), it does not prevent interrupts being serviced due to bus stalling.

Figure 17-46. EPI Write FIFO Count (EPIWFIFOCNT) Register [offset 0x204]

31	3	2	0
Reserved		WTAV	
R-0x0000.000		R-0x4	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17-32. EPI Write FIFO Count (EPIWFIFOCNT) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2-0	WTAV		Available Write Transactions The number of write transactions available in the WFIFO. When clear, a write is stalled waiting for a slot to become free (from a preceding write completing).

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.20 EPI DMA Transmit Count (EPIDMATXCNT) Register, offset 0x208

This register is used to program the total number of transfers (byte, halfword or word) by the µDMA to WRFIFO. As each transfer is processed by the EPI, the TXCNT bit field value is decreased by 1. When TXCNT = 0, the EPI's uDMA request signal is de-asserted

Figure 17-47. EPI DMA Transmit Count (EPIDMATXCNT) Register [offset 0x208]

31	16	15	3	0
Reserved		TXCNT		
R-0x0000		R/W-000		

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-33. EPI DMA Transmit Count (EPIDMATXCNT) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	TXCNT		DMA Count This field is used to program the total number of transfers (byte, halfword or word) from the µDMA to the EPI WRFIFO.

17.11.21 EPI Interrupt Mask (EPIIM) Register, 0x210

This register is the interrupt mask set or clear register. For each interrupt source (read, write, and error), a mask value of 1 allows the interrupt source to trigger an interrupt to the interrupt controller; a mask value of 0 prevents the interrupt source from triggering an interrupt.

Figure 17-48. EPI Interrupt Mask (EPIIM) Register [offset 0x210]

31	16
Reserved	
R-0x000	

15	5	4	3	2	1	0
Reserved		DMAWRIM	DMARDIM	WRIM	RDIM	ERRIM
R-0x000		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-34. EPI Interrupt Mask (EPIIM) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4	DMAWRIM		Write uDMA Interrupt Mask
		0	DMAWRRIS in the EPIRIS register is masked and does not cause an interrupt
		1	DMAWRRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller
3	DMARDIM		Read uDMA Interrupt Mask
		0	DMARDRIS in the EPIRIS register is masked and does not cause an interrupt
		1	DMARDRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller
2	WRIM		Write Interrupt Mask
		0	WRRIS in the EPIRIS register is masked and does not cause an interrupt.
		1	WRRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller.
1	RDIM		Read Interrupt Mask
		0	RDRIS in the EPIRIS register is masked and does not cause an interrupt.
		1	RDRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-34. EPI Interrupt Mask (EPIIM) Register Field Descriptions (continued)

Bit	Field	Value	Description
0	ERRIM		Error Interrupt Mask
		0	ERRIS in the EPIRIS register is masked and does not cause an interrupt.
		1	ERRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller.

17.11.22 EPI Raw Interrupt Status (EPIRIS) Register, offset 0x214

This is the raw interrupt status register. On a read, it gives the current state of each interrupt source. A write has no effect.

Note that raw status for read and write is set or cleared based on FIFO fullness as controlled by EPIFIFOLVL.

Raw status for error is held until the error is cleared by writing to the EPIEISC register.

Figure 17-49. EPI Raw Interrupt Status (EPIRIS) Register [offset 0x214]

31	16
Reserved	
R-0x000	

15	5	4	3	2	1	0
Reserved		DMAWRRIS	DMARDRIS	WRRIS	RDRIS	ERRRIS
R-0x000		R-0	R-0	R-1	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-35. EPI Raw Interrupt Status (EPIRIS) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4	DMAWRRIS		Write uDMA Raw Interrupt Status
		0	The write uDMA has not completed.
		1	The write uDMA has completed.
3	DMARDRIS		Read uDMA Raw Interrupt Status
		0	The read uDMA has not completed
		1	The read uDMA has completed
2	WRRIS		Write Raw Interrupt Status This bit is cleared when the level in the WFIFO is above the trigger point programmed by the WRFIFO field.
		0	The number of available entries in the WFIFO is above the range specified by the WRFIFO field in the EPIFIFOLVL register.
		1	The number of available entries in the WFIFO is within the range specified by the trigger level (the WRFIFO field in the EPIFIFOLVL register).
1	RDRIS		Read Raw Interrupt Status This bit is cleared when the level in the NBRFIFO is below the trigger point programmed by the RDFIFO field
		0	The number of valid entries in the NBRFIFO is below the trigger range specified by the RDFIFO field in the EPIFIFOLVL register
		1	The number of valid entries in the NBRFIFO is in the trigger range specified by the RDFIFO field in the EPIFIFOLVL register

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-35. EPI Raw Interrupt Status (EPIRIS) Register Field Descriptions (continued)

Bit	Field	Value	Description
0	ERRRIS		Error Raw Interrupt Status The error interrupt occurs in the following situations: • WFIFO Full. For a full WFIFO to generate an error interrupt, the WFERR bit in the EPIFIFOLVL register must be set. • Read Stalled. For a stalled read to generate an error interrupt, the RSERR bit in the EPIFIFOLVL register must be set. • Timeout. If the MAXWAIT field in the EPIGPCFG register is configured to a value other than 0, a timeout error occurs when iRDY or XFIFO not-ready signals hold a transaction for more than the count in the MAXWAIT field. To determine which error occurred, read the status of the EPI Error Interrupt Status and Clear (EPIEISC) register. This bit is cleared by writing a 1 to the bit in the EPIEISC register that caused the interrupt.
		0	An error has not occurred.
		1	A WFIFO Full, a Read Stalled, or a Timeout error has occurred.

17.11.23 EPI Masked Interrupt Status (EPIMIS) Register, offset 0x218

This register is the masked interrupt status register. On read, it gives the current state of each interrupt source (read, write, and error) after being masked via the EPIIM register. A write has no effect.

The values returned are the ANDing of the EPIIM and EPIRIS registers. If a bit is set in this register, the interrupt is sent to the interrupt controller.

Figure 17-50. EPI Masked Interrupt Status (EPIMIS) Register [offset 0x218]

31							16
Reserved							
R-0x0000.000							

15	5	4	3	2	1	0
Reserved		DMAWRMIS	DMARDMIS	WRMIS	RDMIS	ERRMIS
R-0x0000.000		R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-36. EPI Masked Interrupt Status (EPIMIS) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4	DMAWRMIS		Write uDMA Masked Interrupt Status This bit is cleared by writing a 1 to the DMAWRIC bit in the EPIEISC register.
		0	The write uDMA has not completed or the interrupt is masked.
		1	The write uDMA has completed and the DMAWRIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller.
3	DMARDMIS		Read uDMA Masked Interrupt Status This bit is cleared by writing a 1 to the DMARDIC bit in the EPIEISC register.
		0	The read uDMA has not completed or the interrupt is masked.
		1	The read uDMA has completed and the DMAWRIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller.
2	WRMIS		Write Masked Interrupt Status
		0	The number of available entries in the WFIFO is above the range specified by the trigger level or the interrupt is masked.
		1	The number of available entries in the WFIFO is within the range specified by the trigger level (the WRFIFO field in the EPIFIFOLVL register) and the WRIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-36. EPI Masked Interrupt Status (EPIMIS) Register Field Descriptions (continued)

Bit	Field	Value	Description
1	RDMIS		Read Masked Interrupt Status
		0	The number of valid entries in the NBRFIFO is below the range specified by the trigger level or the interrupt is masked.
		1	The number of valid entries in the NBRFIFO is within the range specified by the trigger level (the RDFIFO field in the EPIFIFOLVL register) and the RDIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller.
0	ERRMIS		Error Masked Interrupt Status
		0	An error has not occurred or the interrupt is masked.
		1	A WFIFO Full, a Read Stalled, or a Timeout error has occurred and the ERIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller.

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.24 EPI Error Interrupt Status and Clear (EPIEISC) Register, offset 0x21C

This register is used to clear a pending error interrupt. Clearing any defined bit in the EPIEISC has no effect; setting a bit clears the error source and the raw error returns to 0. it indicates that the ERRRIS bit in the EPIRIS register is set and an EPI controller error is sent to the interrupt controller if the ERIM bit in the EPIIM register is set. If any of bits [2:0] are written as 1, the register bit being written to, as well as the ERRIS bit in the EPIRIS register and the ERIM bit in the EPIIM register are cleared. If the DMAWRIC or DMARDIC bit in this register is set, then the corresponding bit in the EPIRIS and EPIMIS register is cleared. Note that writing to this register and reading back immediately (pipelined by the processor) returns the old register contents. One cycle is needed between write and read.his register is used to clear a pending error interrupt. If any of these bits are set, the ERRRIS bit in the EPIRIS register is set, and an EPI controller error is sent to the interrupt controller if the ERIM bit in the EPIIM register is set. Clearing any defined bit has no effect; setting a bit clears the error source and the raw error returns to 0. Note that writing to this register and reading back immediately (pipelined by the processor) returns the old register contents. One cycle is needed between write and read.

Figure 17-51. EPI Error Interrupt Status and Clear (EPIEISC) Register [offset 0x21C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0x0000.000															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											DMAWRIC	DMARDIC	WTFULL	RSTALL	TOUT
R-0x0000.000											W1C-0	W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-37. EPI Error Interrupt Status and Clear (EPIEISC) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4	DMAWRIC		Write uDMA Interrupt Clear
		0	
		1	Writing a 1 to this bit clears the DMAWRRIS bit in the EPIRIS register and the DMAWRMIS bit in the EPIMIS register.
3	DMARDIC		Read uDMA Interrupt Clear
		0	
		1	Writing a 1 to this bit clears the DMARDRIS bit in the EPIRIS register and the DMARDMIS bit in the EPIMIS register.
2	WTFULL		Write FIFO Full Error Writing a 1 to this bit clears it, as well as the ERRRIS and ERIM bits.
		0	The WFERR bit is not enabled or no writes are stalled.
		1	The WFERR bit is enabled and a write is stalled due to the WFIFO being full.
1	RSTALL		Read Stalled Error Writing a 1 to this bit clears it, as well as as the ERRRIS and ERIM bits.
		0	The RSERR bit is not enabled pr no pending reads are stalled.
		1	The RSERR bit is enabled and a pending read is stalled due to writes in the WFIFO.
0	TOUT		Timeout Error This bit is the timeout error source. The timeout error occurs when the iRDY or XFIFO not-ready signals hold a transaction for more than the count in the MAXWAIT field (when not 0). Writing a 1 to bit this clears it, as well as the ERRRIS and ERIM bits.
		0	No timeout error has occurred.
		1	A timeout error has occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.25 EPI Host-Bus 8 Configuration 3 (EPIHB8CFG3), offset 0x308

NOTE: The MODE field in the EPICFG register configures whether EPI Host Bus mode is enabled. For EPIHB8CFG3 to be valid, the MODE field must be 0x2.

Figure 17-52. EPI Host-Bus 8 Configuration 3 Register (EPIHB8CFG3) [offset 0x308]

31	22	21	20	19	18	16
Reserved		WRHIGH	RDHIGH	ALEHIGH	Reserved	
R-0		R/W-0	R/W-0	R/W-1	R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		WRWS		RDWS		Reserved		MODE	
R-0		R/W-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-38. EPI Host-Bus 8 Configuration 3 Register (EPIHB8CFG3) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved	0	Reserved
21	WRHIGH		$\overline{CS2}$ WRITE Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB8CFG2 .
		0	The WRITE strobe for $\overline{CS2}$ accesses is $\overline{WR}$ (active Low).
		1	The WRITE strobe for $\overline{CS2}$ accesses is WR (active High).
20	RDHIGH		$\overline{CS2}$ READ Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB8CFG2.
		0	The READ strobe for $\overline{CS2}$ accesses is $\overline{RD}$ (active Low).
		1	The READ strobe for $\overline{CS2}$ accesses is RD (active High).
19	ALEHIGH		$\overline{CS2}$ ALE Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB8CFG2.
		0	The address latch strobe for $\overline{CS2}$ accesses is $\overline{ADV}$ (active Low).
		1	The address latch strobe for $\overline{CS2}$ accesses is ALE (active High).
18-8	Reserved		Reserved
7-6	WRWS		$\overline{CS2}$ Write Wait States This field is used in conjunction with the EPIBAUD register. This field adds wait states to the data phase of $\overline{CS2}$ accesses (the address phase is not affected). The effect is to delay the rising edge of $\overline{WR}$ (or the falling edge of WR). Each wait state adds two EPI clock cycles to the access time. The WRWSM bit in the EPIHB8TIME3 register can decrease the number of wait states by one EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register. This field is not applicable in BURST mode.
		0x0	Active $\overline{WR}$n is 2 EPI clocks
		0x1	Active $\overline{WR}$ is 4 EPI clocks
		0x2	Active $\overline{WR}$ is 6 EPI clocks
		0x3	Active $\overline{WR}$ is 8 EPI clocks
5-4	RDWS		$\overline{CS2}$ Read Wait States This field is used in conjunction with the EPIBAUD register. This field adds wait states to the data phase of $\overline{CS2}$ accesses (the address phase is not affected). The effect is to delay the rising edge of $\overline{RD}$/$\overline{Oe}$ (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the EPIHB8TIME3 register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register. This field is not applicable in BURST mode.
		0x0	Active $\overline{RD}$ is 2 EPI clocks
		0x1	Active $\overline{RD}$ is 4 EPI clocks
		0x2	Active $\overline{RD}$ is 6 EPI clocks
		0x3	Active $\overline{RD}$ is 8 EPI clocks
3-2	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-38. EPI Host-Bus 8 Configuration 3 Register (EPIHB8CFG3) Field Descriptions (continued)

Bit	Field	Value	Description
1-0	MODE		$\overline{CS2}$ Host Bus Sub-Mode This field determines which Host Bus 8 sub-mode to use for $\overline{CS2}$ in multiple chip-select mode. Sub-mode use is determined by the connected external peripheral. See Table 17-6 for information on how this bit field affects the operation of the EPI signals. **Note:** The CSBAUD bit must be set to enable this $\overline{CS2}$ MODE field. If CSBAUD is clear, all chip selects use the MODE configuration defined in the EPIHB8CFG register.
		0x0	ADMUX – AD[7:0] Data and Address are muxed.
		0x1	ADNONMUX – D[7:0] Data and address are separate.
		0x2	Continuous Read - D[7:0] This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of $\overline{OE}$ strobing.
		0x3	Reserved

17.11.26 EPI Host-Bus 16 Configuration 3 (EPIHB16CFG3), offset 0x308

NOTE: The MODE field in the EPICFG register determines which configuration is enabled. For EPIHB16CFG3 to be valid, the MODE field must be 0x3.

Figure 17-53. EPI Host-Bus 16 Configuration 3 Register (EPIHB16CFG3) [offset 0x308]

31	22	21	20	19	18	17	16
Reserved		WRHIGH	RDHIGH	ALEHIGH	Reserved		BURST
R-0		R/W-0	R/W-0	R/W-1		R-0	R/W-0

15	8	7	6	5	2	1	0
Reserved		WRWS		Reserved		MODE	
R-0		R/W-0		R/W-0	R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17-39. EPI Host-Bus 16 Configuration 3 Register (EPIHB16CFG3) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved		Reserved
21	WRHIGH		$\overline{CS2}$ WRITE Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB16CFG2 .
		0	The WRITE strobe for $\overline{CS2}$ accesses is $\overline{WR}$ (active Low).
		1	The WRITE strobe for $\overline{CS2}$ accesses is WR (active High).
20	RDHIGH		$\overline{CS2}$ READ Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB16CFG2.
		0	The READ strobe for $\overline{CS2}$ accesses is $\overline{RD}$ (active Low).
		1	The READ strobe for $\overline{CS2}$ accesses is RD (active High).
19	ALEHIGH		$\overline{CS2}$ ALE Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB16CFG2.
		0	The address latch strobe for $\overline{CS2}$ accesses is $\overline{ADV}$ (active Low).
		1	The address latch strobe for $\overline{CS2}$ accesses is ALE (active High).
18-17	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-39. EPI Host-Bus 16 Configuration 3 Register (EPIHB16CFG3) Field Descriptions (continued)

Bit	Field	Value	Description
16	BURST		$\overline{CS2}$ Burst Mode Burst mode must be used with an ALE, which is configured by programming the CSCFG and CSCFGEXT fields in the EPIHB16CFG2 register. Burst mode must be used in ADMUX, which is set by the MODE field in EPIHB16CFG3. **Note:** Burst mode is optimized for word-length accesses.
		0	Burst mode is disabled
		1	Burst mode is enabled for $\overline{CS2}$
15-8	Reserved		Reserved
7-6	WRWS		$\overline{CS2}$ Write Wait States This field is used in conjunction with the EPIBAUD register. This field adds wait states to the data phase of $\overline{CS2}$ accesses (the address phase is not affected). The effect is to delay the rising edge of $\overline{WR}$ (or the falling edge of WR). Each wait state adds two EPI clock cycles to the access time. The WRWSM bit in the EPIHB16TIME3 register can decrease the number of wait states by one EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB16CFG2 register. This field is not applicable in BURST mode.
		0x0	Active $\overline{WR}$ is 2 EPI clocks
		0x1	Active $\overline{WR}$ is 4 EPI clocks
		0x2	Active $\overline{WR}$ is 6 EPI clocks
		0x3	Active $\overline{WR}$ is 8 EPI clocks
5-2	Reserved		Reserved
1-0	MODE		$\overline{CS2}$ Host Bus Sub-Mode This field determines which Host Bus 8 sub-mode to use for $\overline{CS2}$ in multiple chip-select mode. Sub-mode use is determined by the connected external peripheral. See Table 17-7 for information on how this bit field affects the operation of the EPI signals. **Note:** The CSBAUD bit must be set to enable this $\overline{CS2}$ MODE field. If CSBAUD is clear, all chip selects use the MODE configuration defined in the EPIHB16CFG register.
		0x0	ADMUX – AD[15:0] Data and Address are muxed.
		0x1	ADNONMUX – D[15:0] Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.
		0x2	Continuous Read - D[15:0] This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of $\overline{OE}$ strobing. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available.
		0x3	Reserved

17.11.27 EPI Host-Bus 8 Configuration 4 (EPIHB8CFG4), offset 0x30C

NOTE: The MODE field in the EPICFG register configures whether EPI Host Bus mode is enabled. For EPIHB8CFG4 to be valid, the MODE field must be 0x2.

Figure 17-54. EPI Host-Bus 8 Configuration 4 Register (EPIHB8CFG4) [offset 0x30C]

31	22	21	20	19	18	16
Reserved		WRHIGH	RDHIGH	ALEHIGH	Reserved	
R-0		R/W-0	R/W-0	R/W-1	R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		WRWS		RDWS		Reserved		MODE	
R-0		R/W-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-40. EPI Host-Bus 8 Configuration 4 Register (EPIHB8CFG4) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved		Reserved
21	WRHIGH		$\overline{CS3}$ WRITE Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB8CFG2 .
		0	The WRITE strobe for $\overline{CS3}$ accesses is $\overline{WR}$ (active Low).
		1	The WRITE strobe for $\overline{CS3}$ accesses is WR (active High).
20	RDHIGH		$\overline{C32}$ READ Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB8CFG2.
		0	The READ strobe for $\overline{CS3}$ accesses is $\overline{RD}$ (active Low).
		1	The READ strobe for $\overline{CS3}$ accesses is RD (active High).
19	ALEHIGH		$\overline{CS3}$ ALE Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB8CFG2.
		0	The address latch strobe for $\overline{CS3}$ accesses is $\overline{ADV}$ (active Low).
		1	The address latch strobe for $\overline{CS3}$ accesses is ALE (active High).
18-8	Reserved		Reserved
7-6	WRWS		$\overline{CS3}$ Write Wait States This field is used in conjunction with the EPIBAUD register. This field adds wait states to the data phase of $\overline{CS3}$ accesses (the address phase is not affected). The effect is to delay the rising edge of $\overline{WR}$ (or the falling edge of WR). Each wait state adds two EPI clock cycles to the access time. The WRWSM bit in the EPIHB8TIME4 register can decrease the number of wait states by one EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register. This field is not applicable in BURST mode.
		0x0	Active $\overline{WR}$ is 2 EPI clocks
		0x1	Active $\overline{WR}$ is 4 EPI clocks
		0x2	Active $\overline{WR}$ is 6 EPI clocks
		0x3	Active $\overline{WR}$ is 8 EPI clocks
5-4	RDWS		$\overline{CS3}$ Read Wait States This field is used in conjunction with the EPIBAUD register. This field adds wait states to the data phase of $\overline{CS3}$ accesses (the address phase is not affected). The effect is to delay the rising edge of $\overline{RD}$/$\overline{OE}$ (or the falling edge of $\overline{RD}$). Each wait state adds two EPI clock cycles to the access time. The RDWSM bit in the EPIHB8TIME4 register can decrease the number of wait states by one EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB8CFG2 register. This field is not applicable in BURST mode.
		0x0	Active $\overline{RD}$ is 2 EPI clocks
		0x1	Active $\overline{RD}$ is 4 EPI clocks
		0x2	Active$\overline{RD}$ is 6 EPI clocks
		0x3	Active $\overline{RD}$ is 8 EPI clocks
3-2	Reserved		Reserved
1-0	MODE		$\overline{CS3}$ Host Bus Sub-Mode This field determines which Host Bus 8 sub-mode to use for $\overline{CS3}$ in multiple chip-select mode. Sub-mode use is determined by the connected external peripheral. See Table 17-6 for information on how this bit field affects the operation of the EPI signals. **Note:** The CSBAUD bit must be set to enable this $\overline{CS3}$ MODE field. If CSBAUD is clear, all chip selects use the MODE configuration defined in the EPIHB8CFG register.
		0x0	ADMUX – AD[7:0] Data and Address are muxed.
		0x1	ADNONMUX – D[7:0] Data and address are separate.
		0x2	Continuous Read - D[7:0] This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of $\overline{OE}$ strobing.
		0x3	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.28 EPI Host-Bus 16 Configuration 4 (EPIHB16CFG4), offset 0x30C

NOTE: The MODE field in the EPICFG register determines which configuration is enabled. For EPIHB16CFG4 to be valid, the MODE field must be 0x3.

Figure 17-55. EPI Host-Bus 16 Configuration 4 Register (EPIHB16CFG4) [offset 0x30C]

31	22	21	20	19	18	17	16
Reserved		WRHIGH	RDHIGH	ALEHIGH	Reserved		BURST
R-0		R/W-0	R/W-0	R/W-1	R-0		R/W-0

15	8	7	6	5	2	1	0
Reserved		WRWS		Reserved			MODE
R-0		R/W-0		R-0			R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-41. EPI Host-Bus 16 Configuration 4 Register (EPIHB16CFG4) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved		Reserved
21	WRHIGH		$\overline{CS3}$ WRITE Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB16CFG2 .
		0	The WRITE strobe for $\overline{CS3}$ accesses is $\overline{WR}$ (active Low).
		1	The WRITE strobe for $\overline{CS3}$ accesses is WR (active High).
20	RDHIGH		$\overline{CS3}$ READ Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB16CFG2.
		0	The READ strobe for $\overline{CS3}$ accesses is $\overline{RD}$ (active Low).
		1	The READ strobe for $\overline{CS3}$ accesses is RD (active High).
19	ALEHIGH		$\overline{CS3}$ ALE Strobe Polarity This field is used if the CSBAUD bit is enabled in EPIHB16CFG2.
		0	The address latch strobe for $\overline{CS3}$ accesses is $\overline{ADV}$ (active Low).
		1	The address latch strobe for $\overline{CS3}$ accesses is ALE (active High).
18-17	Reserved		Reserved
16	BURST		$\overline{CS3}$ Burst Mode Burst mode must be used with an ALE, which is configured by programming the CSCFG and CSCFGEXT fields in the EPIHB16CFG2 register. Burst mode must be used in ADMUX, which is set by the MODE field in EPIHB16CFG4. **Note:** Burst mode is optimized for word-length accesses.
		0	Burst mode is disabled
		1	Burst mode is enabled for $\overline{CS3}$
15-8	Reserved		Reserved
7-6	WRWS		$\overline{CS3}$ Write Wait States This field is used in conjunction with the EPIBAUD register. This field adds wait states to the data phase of $\overline{CS3}$ accesses (the address phase is not affected). The effect is to delay the rising edge of $\overline{WR}$ (or the falling edge of $\overline{WR}$. Each wait state adds two EPI clock cycles to the access time. The WRWSM bit in the EPIHB16TIME4 register can decrease the number of wait states by one EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the EPIHB16CFG2 register. This field is not applicable in BURST mode.
		0x0	Active $\overline{WR}$ is 2 EPI clocks
		0x1	Active $\overline{WR}$ is 4 EPI clocks
		0x2	Active $\overline{WR}$ is 6 EPI clocks
		0x3	Active $\overline{WR}$ is 8 EPI clocks
5-2	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-41. EPI Host-Bus 16 Configuration 4 Register (EPIHB16CFG4) Field Descriptions (continued)

Bit	Field	Value	Description
1-0	MODE		$\overline{CS3}$ Host Bus Sub-Mode This field determines which Host Bus 16 sub-mode to use for $\overline{CS3}$ in multiple chip-select mode. Sub-mode use is determined by the connected external peripheral. See Table 17-7 for information on how this bit field affects the operation of the EPI signals. **Note:** The CSBAUD bit must be set to enable this $\overline{CS3}$ MODE field. If CSBAUD is clear, all chip selects use the MODE configuration defined in the EPIHB16CFG register.
		0x0	ADMUX – AD[15:0] Data and Address are muxed.
		0x1	ADNONMUX – D[15:0] Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.
		0x2	Continuous Read - D[15:0] This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of $\overline{OE}$ strobing. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available.
		0x3	Reserved

17.11.29 EPI Host-Bus 8 Timing Extension (EPIHB8TIME), offset 0x310

NOTE: : The MODE field in the EPICFG register determines which configuration is enabled. For EPIHB8TIME to be valid, the MODE field must be 0x2.

Figure 17-56. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME) [offset 0x310]

31	26	25	24	23	16
Reserved		IRDYDLY		Reserved	
R-0		R/W-0		R-0	

15	14	13	12	11	5	4	3	1	0
Reserved		CAPWIDTH		Reserved		WRWSM	Reserved		RDWSM
R-0		R/W-0x2		R-0		R/W-0	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-42. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25-24	IRDYDLY		$\overline{CS0}$ Input Ready Delay
		0x0	Reserved
		0x1	Stall begins one EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
		0x2	Stall begins two EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock
		0x3	Stall begins three EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
23-14	Reserved		Reserved
13-12	CAPWIDTH		$\overline{CS0}$ Inter-transfer Capture Width Controls the delay between Host-Bus transfers
		0x0	Reserved
		0x1	1 EPI clock
		0x2	2 EPI clock
		0x3	Reserved
11-5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-42. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME) Field Descriptions (continued)

Bit	Field	Value	Description
4	WRWSM		Write Wait State Minus One This bit is used with the WRWS field in EPIHB8CFG. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB8CFG register
		1	Wait state value is now: WRWS - 1 WRWS field is programmed in EPIHB8CFG
3-1	Reserved		Reserved
0	RDWSM		Read Wait State Minus One Used with RDWS field in the EPIHB8CFG register. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB8CFG
		1	Wait state value is now: RDWS - 1 RDWS field is programmed in EPIHB8CFG

17.11.30 EPI Host-Bus 16 Timing Extension (EPIHB16TIME), offset 0x310

NOTE: : The MODE field in the EPICFG register determines which configuration is enabled. For EPIHB16TIME to be valid, the MODE field must be 0x3.

Figure 17-57. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME) [offset 0x310]

31	26	25	24	23	16
Reserved		IRDYDLY		Reserved	
R-0		R/W-0		R-0	

15	14	13	12	11	5	4	3	1	0
Reserved		CAPWIDTH		Reserved		WRWSM	Reserved		RDWSM
R-0		R/W-0x2		R-0		R/W-0	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-43. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25-24	IRDYDLY		$\overline{CS0}$ Input Ready Delay
		0x0	Reserved
		0x1	Stall begins one EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
		0x2	Stall begins two EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock
		0x3	Stall begins three EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
23-14	Reserved		Reserved
13-12	CAPWIDTH		$\overline{CS0}$ Inter-transfer Capture Width Controls the delay between Host-Bus transfers
		0x0	Reserved
		0x1	1 EPI clock
		0x2	2 EPI clock
		0x3	Reserved
11-5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-43. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME) Field Descriptions (continued)

Bit	Field	Value	Description
4	WRWSM		$\overline{CS0}$ - Write Wait State Minus One This bit is used with the WRWS field in EPIHB16CFG. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB16CFG register
		1	Wait state value is now: WRWS - 1 WRWS field is programmed in EPIHB16CFG
3-1	Reserved		Reserved
0	RDWSM		$\overline{CS0}$ - Read Wait State Minus One Used with RDWS field in the EPIHB16CFG register. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB16CFG
		1	Wait state value is now: RDWS - 1 RDWS field is programmed in EPIHB16CFG

17.11.31 EPI Host-Bus 8 Timing Extension (EPIHB8TIME2), offset 0x314

NOTE: The MODE field in the EPICFG register determines which configuration is enabled. For EPIHB8TIME2 to be valid, the MODE field must be 0x2.

Figure 17-58. EPI Host-Bus 8 Timing Extension (EPIHB8TIME2) Register [offset 0x314]

31	26	25	24	23	16
Reserved		IRDYDLY		Reserved	
R-0		R/W-0		R-0	

15	14	13	12	11	5	4	3	1	0
Reserved		CAPWIDTH		Reserved		WRWSM	Reserved		RDWSM
R-0		R/W-0x2		R-0		R/W-0	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-44. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME2) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25-24	IRDYDLY		$\overline{CS1}$ Input Ready Delay
		0x0	Reserved
		0x1	Stall begins one EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
		0x2	Stall begins two EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock
		0x3	Stall begins three EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
23-14	Reserved		Reserved
13-12	CAPWIDTH		$\overline{CS1}$ Inter-transfer Capture Width Controls the delay between Host-Bus transfers
		0x0	Reserved
		0x1	1 EPI clock
		0x2	2 EPI clock
		0x3	Reserved
11-5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-44. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME2) Field Descriptions (continued)

Bit	Field	Value	Description
4	WRWSM		$\overline{CS1}$ Write Wait State Minus One This bit is used with the WRWS field in EPIHB8CFG2. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB8CFG2 register
		1	Wait state value is now: WRWS - 1 WRWS field is programmed in EPIHB8CFG2
3-1	Reserved		Reserved
0	RDWSM		$\overline{CS1}$ Read Wait State Minus One Used with RDWS field in the EPIHB8CFG2 register. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB8CFG2
		1	Wait state value is now: RDWS - 1 RDWS field is programmed in EPIHB8CFG2

17.11.32 EPI Host-Bus 16 Timing Extension (EPIHB16TIME2), offset 0x314

NOTE: : The MODE field in the EPICFG register determines which configuration is enabled. For EPIHB16TIME2 to be valid, the MODE field must be 0x3.

Figure 17-59. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME2) [offset 0x314]

31	26	25	24	23	16
Reserved		IRDYDLY		Reserved	
R-0		R/W-0		R-0	

15	14	13	12	11	5	4	3	1	0
Reserved		CAPWIDTH		Reserved		WRWSM	Reserved		RDWSM
R-0		R/W-0x2		R-0		R/W-0	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-45. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME2) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25-24	IRDYDLY		$\overline{CS1}$ Input Ready Delay
		0x0	Reserved
		0x1	Stall begins one EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
		0x2	Stall begins two EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock
		0x3	Stall begins three EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
23-14	Reserved		Reserved
13-12	CAPWIDTH		$\overline{CS1}$ Inter-transfer Capture Width Controls the delay between Host-Bus transfers
		0x0	Reserved
		0x1	1 EPI clock
		0x2	2 EPI clock
		0x3	Reserved
11-5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-45. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME2) Field Descriptions (continued)

Bit	Field	Value	Description
4	WRWSM		$\overline{CS1}$ Write Wait State Minus One This bit is used with the WRWS field in EPIHB16CFG2. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB16CFG2 register
		1	Wait state value is now: WRWS - 1 WRWS field is programmed in EPIHB16CFG 2
3-1	Reserved		Reserved
0	RDWSM		$\overline{CS1}$ Read Wait State Minus One Used with RDWS field in the EPIHB16CFG2 register. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB16CFG2
		1	Wait state value is now: RDWS - 1 RDWS field is programmed in EPIHB16CFG2

17.11.33 EPI Host-Bus 8 Timing Extension (EPIHB8TIME3), offset 0x318

NOTE: The MODE field in the EPICFG register determines which configuration is enabled. For EPIHB8TIME3 to be valid, the MODE field must be 0x2.

Figure 17-60. EPI Host-Bus 8 Timing Extension (EPIHB8TIME3) Register [offset 0x318]

31	26	25	24	23	16
Reserved		IRDYDLY		Reserved	
R-0		R/W-0		R-0	

15	14	13	12	11	5	4	3	1	0
Reserved		CAPWIDTH		Reserved		WRWSM	Reserved		RDWSM
R-0		R/W-0x2		R-0		R/W-0	R-0		

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-46. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME3) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25-24	IRDYDLY		$\overline{CS1}$ Input Ready Delay
		0x0	Reserved
		0x1	Stall begins one EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
		0x2	Stall begins two EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock
		0x3	Stall begins three EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
23-14	Reserved		Reserved
13-12	CAPWIDTH		$\overline{CS1}$ Inter-transfer Capture Width Controls the delay between Host-Bus transfers
		0x0	Reserved
		0x1	1 EPI clock
		0x2	2 EPI clock
		0x3	Reserved
11-5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-46. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME3) Field Descriptions (continued)

Bit	Field	Value	Description
4	WRWSM		$\overline{CS1}$ Write Wait State Minus One This bit is used with the WRWS field in EPIHB8CFG2. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB8CFG2 register
		1	Wait state value is now: WRWS - 1 WRWS field is programmed in EPIHB8CFG2
3-1	Reserved		Reserved
0	RDWSM		$\overline{CS1}$ Read Wait State Minus One Used with RDWS field in the EPIHB8CFG2 register. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB8CFG2
		1	Wait state value is now: RDWS - 1 RDWS field is programmed in EPIHB8CFG2

17.11.34 EPI Host-Bus 16 Timing Extension (EPIHB16TIME3) Register, offset 0x318

NOTE: The MODE field in the EPICFG register determines which configuration is enabled. For EPIHB16TIME3 to be valid, the MODE field must be 0x3.

Figure 17-61. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME3) [offset 0x318]

31	26	25	24	23	16
Reserved		IRDYDLY		Reserved	
R-0		R/W-0		R-0	

15	14	13	12	11	5	4	3	1	0
Reserved		CAPWIDTH		Reserved		WRWSM	Reserved		RDWSM
R-0		R/W-0x2		R-0		R/W-0	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-47. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME3) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25-24	IRDYDLY		$\overline{CS2}$ Input Ready Delay
		0x0	Reserved
		0x1	Stall begins one EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
		0x2	Stall begins two EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock
		0x3	Stall begins three EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
23-14	Reserved		Reserved
13-12	CAPWIDTH		$\overline{CS2}$ Inter-transfer Capture Width Controls the delay between Host-Bus transfers
		0x0	Reserved
		0x1	1 EPI clock
		0x2	2 EPI clock
		0x3	Reserved
11-5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-47. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME3) Field Descriptions (continued)

Bit	Field	Value	Description
4	WRWSM		$\overline{CS2}$ Write Wait State Minus One This bit is used with the WRWS field in EPIHB16CFG3. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB16CFG3 register
		1	Wait state value is now: WRWS - 1 WRWS field is programmed in EPIHB16CFG3
3-1	Reserved		Reserved
0	RDWSM		$\overline{CS2}$ Read Wait State Minus One Used with RDWS field in the EPIHB16CFG3 register. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB16CFG3
		1	Wait state value is now: RDWS - 1 RDWS field is programmed in EPIHB16CFG3

17.11.35 EPI Host-Bus 8 Timing Extension (EPIHB8TIME4) Register, offset 0x31C

NOTE: The MODE field in the EPICFG register determines which configuration is enabled. For EPIHB8TIME4 to be valid, the MODE field must be 0x2.

Figure 17-62. EPI Host-Bus 8 Timing Extension (EPIHB8TIME4) Register [offset 0x31C]

31	26	25	24	23	16
Reserved		IRDYDLY		Reserved	
R-0		R/W-0		R-0	

15	14	13	12	11	5	4	3	1	0
Reserved		CAPWIDTH		Reserved		WRWSM	Reserved		RDWSM
R-0		R/W-0x2		R-0		R/W-0	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-48. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME3) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25-24	IRDYDLY		$\overline{CS3}$ Input Ready Delay
		0x0	Reserved
		0x1	Stall begins one EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
		0x2	Stall begins two EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock
		0x3	Stall begins three EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
23-14	Reserved		Reserved
13-12	CAPWIDTH		$\overline{CS3}$ Inter-transfer Capture Width Controls the delay between Host-Bus transfers
		0x0	Reserved
		0x1	1 EPI clock
		0x2	2 EPI clock
		0x3	Reserved
11-5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-48. EPI Host-Bus 8 Timing Extension Register (EPIHB8TIME3) Field Descriptions (continued)

Bit	Field	Value	Description
4	WRWSM		$\overline{CS3}$ Write Wait State Minus One This bit is used with the WRWS field in EPIHB8CFG4. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB8CFG4 register
		1	Wait state value is now: WRWS - 1 WRWS field is programmed in EPIHB8CFG4
3-1	Reserved		Reserved
0	RDWSM		$\overline{CS3}$ Read Wait State Minus One Used with RDWS field in the EPIHB8CFG4 register. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB8CFG4
		1	Wait state value is now: RDWS - 1 RDWS field is programmed in EPIHB8CFG4

17.11.36 EPI Host-Bus 16 Timing Extension (EPIHB16TIME4) Register, 0x31C

NOTE: The MODE field in the EPICFG register determines which configuration is enabled. For EPIHB16TIME4 to be valid, the MODE field must be 0x3.

Figure 17-63. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME4) [offset 0x31C]

31	26	25	24	23	16
Reserved		IRDYDLY		Reserved	
R-0		R/W-0		R-0	

15	14	13	12	11	5	4	3	1	0
Reserved		CAPWIDTH		Reserved		WRWSM	Reserved		RDWSM
R/W-0x2				R-0		R/W-0	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-49. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME4) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25-24	IRDYDLY		$\overline{CS3}$ Input Ready Delay
		0x0	Reserved
		0x1	Stall begins one EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
		0x2	Stall begins two EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock
		0x3	Stall begins three EPI clocks past iRDY low being sampled on the rising edge of the EPIO clock.
23-14	Reserved		Reserved
13-12	CAPWIDTH		$\overline{CS3}$ Inter-transfer Capture Width Controls the delay between Host-Bus transfers
		0x0	Reserved
		0x1	1 EPI clock
		0x2	2 EPI clock
		0x3	Reserved
11-5	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-49. EPI Host-Bus 16 Timing Extension Register (EPIHB16TIME4) Field Descriptions (continued)

Bit	Field	Value	Description
4	WRWSM		$\overline{CS3}$ Write Wait State Minus One This bit is used with the WRWS field in EPIHB16CFG4. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the in WRWS field in EPIHB16CFG4 register
		1	Wait state value is now: WRWS - 1 WRWS field is programmed in EPIHB16CFG4
3-1	Reserved		Reserved
0	RDWSM		$\overline{CS3}$ Read Wait State Minus One Used with RDWS field in the EPIHB16CFG4 register. This field is not applicable in BURST mode.
		0	No change in the number of wait state clock cycles programmed in the RDWS field of EPIHB16CFG4
		1	Wait state value is now: RDWS - 1 RDWS field is programmed in EPIHB16CFG4

17.11.37 CEPIRTWCFG Register

Figure 17-64. CEPIRTWCFG Register

31	4	3	1	0
KEY		Reserved		GRAB
R/W-0		R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-50. CEPIRTWCFG Register Field Descriptions

Bit	FiKEY	Value	Description
31-4	KEY		This field should have value of “0x9EDCB4A” to enable write into the GRAB bit. During every write to GRAB bit this field must have a valid value, hence the user always has to do a 32-bit write to change the GRAB bit.
3-1	Reserved		Reserved
0	GRAB		Grabs the access for control subsystem (C28x-CPU/C28x-DMA). This bit gets reset to ‘0’ when TIMEOUT bit in CEPISTATUS register is set.
		0	RTW feature is disabled. All the accesses to the external device via EPI are arbitrated based on round robin scheme.
		1	RTW feature is enabled. All the accesses from the Master subsystem (M3/uDMA) to the external device via EPI are stalled until this bit is set to ‘1’.

17.11.38 CEPIRTWCNT Register

Figure 17-65. CEPIRTWCNT Register

31	0
EPIRTWCNT	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-51. CEPIRTWCNT Register Field Descriptions

Bit	FiKEY	Value	Description
31-0	EPIRTWCNT		This is a free running counter which counts up for every cycle of C28x CPU CLK. This counter gets enabled when the GRAB bit in the CEPIRTWCFG registers gets set to ‘1’. This counter gets reset to 0x0 when GRAB bit in CEPIRTWCFG registers get reset to ‘0’.

Copyright © 2012–2019, Texas Instruments Incorporated

17.11.39 CEPIRTWPRD Register

Figure 17-66. CEPIRTWPRD Register

31	0
EPIRTWPRD	
0xFFFF_FFFF	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-52. CEPIRTWPRD Register Field Descriptions

Bit	FiKEY	Value	Description
31-0	EPIRTWPRD		When EPIRTWCNT value becomes equal to value into this field, TIMEOUT occurs and the TIMEOUT status flag in CEPISTATUS register gets set. Program this field based on the application which needs dedicated access to external device from the control subsystem via EPI. Note: Do not try to program this field when the GRAB bit in the CEPIRTWCFG register is set to '1'. Program this field first and then set the GRAB bit.

17.11.40 CEPISTATUS Register

Figure 17-67. CEPISTATUS Register

31	16
Reserved	
R-0	

15	2	3	2	1	0
Reserved		MEMPROTERR		RTWTIMEOUT	
R-0		R/W-1		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-53. CEPISTATUS Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
1	MEMPROTERR		Status bit for Memory Protection violation by control subsystem (C28x-CPU/C28x-DMA).
		0	Memory protection violation has not occurred.
		1	Memory protection violation occurred. C28x-CPU or C28x-DMA tried to write into external memory space which was protected. An interrupt has been generated to C28x CPU.
0	RTWTIMEOUT		Status bit for RTW time out. Writing a '1' to this field clears the status to '0'
		0	TIMEOUT has not occurred.
		1	C28x software didn't release the GRAB with in programmed time period hence TIMEOUT has happened and an interrupt has been generated to C28x CPU.

17.11.41 MEMPROT Register

Figure 17-68. MEMPROT Register

31	16
Reserved	

15	8	7	6	5	4	3	2	1	0
Reserved		CS3PROT		CS2PROT		CS1PROT		CS0PROT	
R-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 17-54. MEMPROT Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-6	CS3PROT	00	CS3 region is not protected. All master (M3/uDMA/C28x/DMA) have RD/WR permission.
		01	Only Master subsystem has WR permission to CS3 region. Control subsystem has only RD permission.
		10	Only control subsystem has WR permission to CS3 region. Master subsystem has only RD permission.
		11	Reserved
5-4	CS2PROT	00	CS2 region is not protected. All master (M3/uDMA/C28x/DMA) have RD/WR permission.
		01	Only Master subsystem has WR permission to CS2 region. Control subsystem has only RD permission.
		10	Only control subsystem has WR permission to CS2 region. Master subsystem has only RD permission.
		11	Reserved
3-2	CS1PROT	00	CS1 region is not protected. All master (M3/uDMA/C28x/DMA) have RD/WR permission
		01	Only Master subsystem has WR permission to CS1 region. Control subsystem has only RD permission
		10	Only control subsystem has WR permission to CS1 region. Master subsystem has only RD permission.
		11	Reserved
1-0	CS0PROT	00	CS0 region is not protected. All master (M3/uDMA/C28x/DMA) have RD/WR permission
		01	Only master subsystem has WR permission to CS0 region. Control subsystem has only RD permission.
		10	Only control subsystem has WR permission to CS0 region. Master subsystem has only RD permission.
		11	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

M3 Universal Serial Bus (USB) Controller

The USB controller operates as a full-speed or low-speed function controller during point-to-point communications with USB Host, Device, or OTG functions. The controller complies with the USB 2.0 standard, which includes SUSPEND and RESUME signaling. Thirty-two endpoints comprised of two hard-wired for control transfers (one endpoint for IN and one endpoint for OUT) plus 30 endpoints defined by firmware along with a dynamic sizable FIFO, support multiple packet queueing. DMA access to the FIFO allows minimal interference from system software. Software-controlled connect and disconnect allows flexibility during USB device start-up. The controller complies with the OTG standard's session request protocol (SRP) and host negotiation protocol (HNP).

Topic **Page**

18.1 Introduction **1280**
18.2 Functional Description **1280**
18.3 Initialization and Configuration **1291**
18.4 Register Map **1292**
18.5 Register Descriptions **1300**

Copyright © 2012–2019, Texas Instruments Incorporated

18.1 Introduction

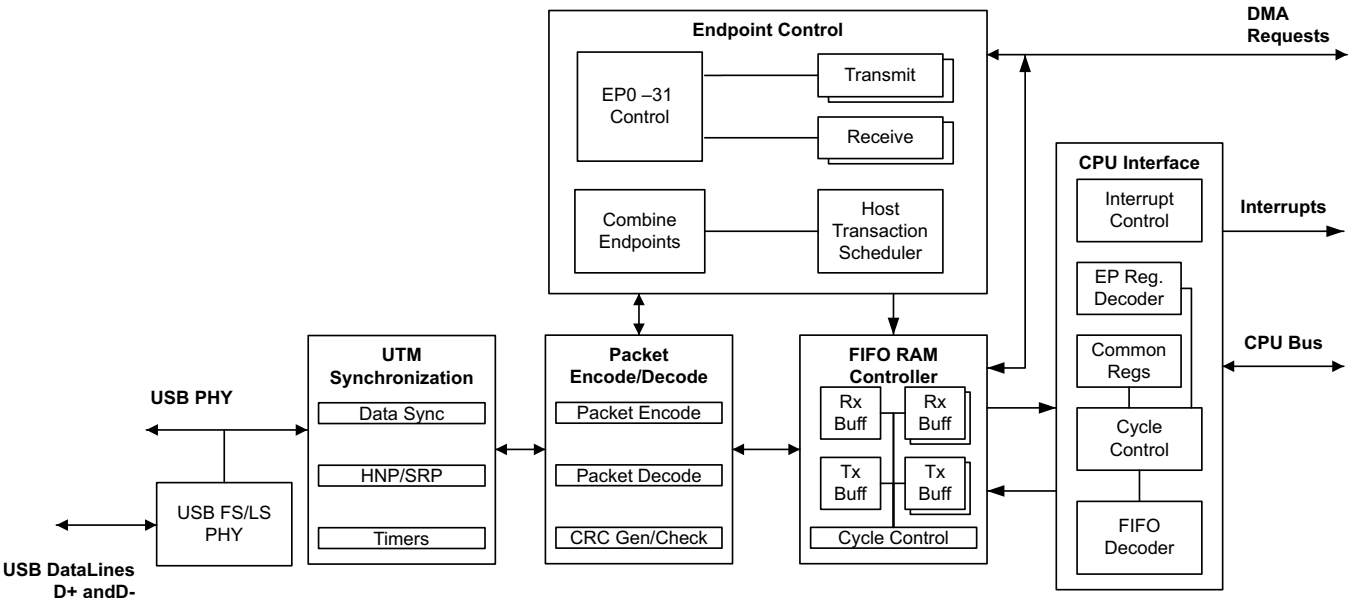
The USB controller operates as a full-speed or low-speed function controller during point-to-point communications with USB Host, Device, or OTG functions.

The USB module has the following features:

- Complies with USB-IF certification standards
- USB 2.0 full-speed (12 Mbps) and low-speed (1.5 Mbps) operation
- Integrated PHY
- Four transfer types: Control, Interrupt, Bulk, and Isochronous
- 32 endpoints
  - One dedicated control IN endpoint and one dedicated control OUT endpoint
  - 15 configurable IN endpoints and 15 configurable OUT endpoints
- Four KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- VBUS droop and valid ID detection and interrupt
- Efficient transfers using direct memory access controller (DMA)
  - Separate channels for transmit and receive for up to three IN endpoints and three OUT endpoints
  - Channel requests asserted when FIFO contains required amount of data

18.1.1 Block Diagram

Figure 18-1. USB Block Diagram



18.2 Functional Description

The USB controller provides full OTG negotiation by supporting both the session request protocol (SRP) and the host negotiation protocol (HNP). The session request protocol allows devices on the B side of a cable to request the A side device turn on VBUS. The host negotiation protocol is used after the initial session request protocol has powered the bus and provides a method to determine which end of the cable will act as the Host controller. When the device is connected to non-OTG peripherals or devices, the controller can detect which cable end was used and provides a register to indicate if the controller should act as the Host or the Device controller. This indication and the mode of operation are handled automatically by the USB controller. This auto-detection allows the system to use a single A/B connector instead of having both A and B connectors in the system and supports full OTG negotiations with other OTG devices.

Copyright © 2012–2019, Texas Instruments Incorporated

In addition, the USB controller provides support for connecting to non-OTG peripherals or Host controllers. The USB controller can be configured to act as either a dedicated Host or Device, in which case, the USB0VBUS and USB0ID signals can be used as GPIOs. However, when the USB controller is acting as a self-powered Device, a GPIO input must be connected to VBUS and configured to generate an interrupt when the VBUS level drops. This interrupt is used to disable the pullup resistor on the USB0DP signal.

Note: When USB is used in the system, the minimum system frequency is 20 MHz.

18.2.1 Operation as a Device

This section describes the USB controller's actions when it is being used as a USB Device. Before the USB controller's operating mode is changed from device to host or host to device, software must reset the USB controller by setting the USB0 bit in the software reset control 2 (SRCR2) register (see the *System Control* chapter). IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and recognition of start of frame (SOF) are all described.

When in device mode, IN transactions are controlled by an endpoint’s transmit interface and use the transmit endpoint registers for the given endpoint. OUT transactions are handled with an endpoint's receive interface and use the receive endpoint registers for the given endpoint. When configuring the size of the FIFOs for endpoints, take into account the maximum packet size for an endpoint.

- **Bulk.** Bulk endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- **Interrupt.** Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- **Isochronous**. Isochronous endpoints are more flexible and can be up to 1023 bytes.
- **Control**. It is also possible to specify a separate control endpoint for a USB device. However, in most cases the USB device should use the dedicated control endpoint on the USB controller’s endpoint 0.

18.2.1.1 Endpoints

When operating as a Device, the USB controller provides two dedicated control endpoints (IN and OUT) and 30 configurable endpoints (15 IN and 15 OUT) that can be used for communications with a Host controller. The endpoint number and direction associated with an endpoint is directly related to its register designation. For example, when the Host is transmitting to endpoint 1, all configuration and data is in the endpoint 1 transmit register interface.

Endpoint 0 is a dedicated control endpoint used for all control transactions to endpoint 0 during enumeration or when any other control requests are made to endpoint 0. Endpoint 0 uses the first 64 bytes of the USB controller's FIFO RAM as a shared memory for both IN and OUT transactions.

The remaining 30 endpoints can be configured as control, bulk, interrupt, or isochronous endpoints. They should be treated as 15 configurable IN and 15 configurable OUT endpoints. The endpoint pairs are not required to have the same type for their IN and OUT endpoint configuration. For example, the OUT portion of an endpoint pair could be a bulk endpoint, while the IN portion of that endpoint pair could be an interrupt endpoint. The address and size of the FIFOs attached to each endpoint can be modified to fit the application's needs.

18.2.1.1.1 IN Transactions as a Device

When operating as a USB device, data for IN transactions is handled through the FIFOs attached to the transmit endpoints. The sizes of the FIFOs for the 15 configurable IN endpoints are determined by the USB Transmit FIFO Start Address (USBTXFIFOADD) register. The maximum size of a data packet that may be placed in a transmit endpoint’s FIFO for transmission is programmable and is determined by the value written to the USB Maximum Transmit Data Endpoint n (USBTXMAXPn) register for that endpoint. The endpoint’s FIFO can also be configured to use double-packet or single-packet buffering. When double-packet buffering is enabled, two data packets can be buffered in the FIFO, which also requires that the FIFO is at least two packets in size. When double-packet buffering is disabled, only one packet can be buffered, even if the packet size is less than half the FIFO size.

Note: The maximum packet size set for any endpoint must not exceed the FIFO size. The USBTXMAXPn register should not be written to while data is in the FIFO, as unexpected results may occur.

Copyright © 2012–2019, Texas Instruments Incorporated

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.

18.2.1.1.2 Out Transactions as a Device

When in Device mode, OUT transactions are handled through the USB controller receive FIFOs. The sizes of the receive FIFOs for the 15 configurable OUT endpoints are determined by the USB Receive FIFO Start Address (USBRXFIFOADD) register. The maximum amount of data received by an endpoint in any packet is determined by the value written to the USB Maximum Receive Data Endpoint n (USBRXMAXPn) register for that endpoint. When double-packet buffering is enabled, two data packets can be buffered in the FIFO. When double-packet buffering is disabled, only one packet can be buffered even if the packet is less than half the FIFO size.

Note: In all cases, the maximum packet size must not exceed the FIFO size.

Single-Packet Buffering

If the size of the receive endpoint FIFO is less than twice the maximum packet size for an endpoint, only one data packet can be buffered in the FIFO and single-packet buffering is required. When a packet is received and placed in the receive FIFO, the RXRDY and FULL bits in the USB Receive Control and Status Endpoint n Low (USBRXCSRL[*n*]) 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 (USBRXCSRH[*n*]) 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

Copyright © 2012–2019, Texas Instruments Incorporated

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 USBRXCSRL[*n*] 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 USBRXCSRL[*n*] 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 USBRXCSRH[*n*] 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.

18.2.1.1.3 Scheduling

The device has no control over the scheduling of transactions as scheduling is determined by the Host controller. The 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.

18.2.1.1.4 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

Copyright © 2012–2019, Texas Instruments Incorporated

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 Ddvice.

18.2.1.1.5 Device Mode Suspend

When no activity has occurred on the USB bus for 3 ms, the USB controller automatically enters SUSPEND mode. If the SUSPEND interrupt has been enabled in the USB Interrupt Enable (USBIE) register, an interrupt is generated at this time. When in SUSPEND mode, the PHY also goes into SUSPEND mode. When RESUME signaling is detected, the USB controller exits SUSPEND mode and takes the PHY out of SUSPEND. If the RESUME interrupt is enabled, an interrupt is generated. The USB controller can also be forced to exit SUSPEND mode by setting the RESUME bit in the USB Power (USBPOWER) register. When this bit is set, the USB controller exits SUSPEND mode and drives RESUME signaling onto the bus. The RESUME bit must be cleared after 10 ms (a maximum of 15 ms) to end RESUME signaling.

To meet USB power requirements, the controller can be put into Deep Sleep mode which keeps the controller in a static state.

18.2.1.1.6 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.

18.2.1.1.7 USB Reset

When the USB controller is in device mode and a RESET condition is detected on the USB bus, the USB controller automatically performs the following actions:

- Clears the USBFADDR register.
- Clears the USB Endpoint Index (USBEPIDX) register.
- Flushes all endpoint FIFOs.
- Clears all control/status registers.
- Enables all endpoint interrupts.
- Generates a RESET interrupt.

Copyright © 2012–2019, Texas Instruments Incorporated

18.2.1.1.8 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.

18.2.2 Operation as a Host

When the USB controller is operating in Host mode, it can either be used for point-to-point communications with another USB device or, when attached to a hub, for communication with multiple devices. Before the USB controller's operating mode is changed from Host to Device or Device to Host, software must reset the USB controller by setting the USB0 bit in the Software Reset Control 2 (SRCR2) register (see the *System Control* chapter). Full-speed and low-speed USB devices are supported, both for point-to-point communication and for operation through a hub. The USB controller automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub. Control, bulk, isochronous, and interrupt transactions are supported. This section describes the USB controller's actions when it is being used as a USB Host. Configuration of IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and RESET are all described.

When in Host mode, IN transactions are controlled by an endpoint’s receive interface. All IN transactions use the receive endpoint registers and all OUT endpoints use the transmit endpoint registers for a given endpoint. As in Device mode, the FIFOs for endpoints should take into account the maximum packet size for an endpoint.

- **Bulk.** Bulk endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- **Interrupt**. Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- **Isochronous.** Isochronous endpoints are more flexible and can be up to 1023 bytes.
- **Control.** It is also possible to specify a separate control endpoint to communicate with a device. However, in most cases the USB controller should use the dedicated control endpoint to communicate with a device’s endpoint 0.

18.2.2.1 Endpoints

The endpoint registers are used to control the USB endpoint interfaces which communicate with Device(s) that are connected. The endpoints consist of a dedicated control IN endpoint, a dedicated control OUT endpoint, 15 configurable OUT endpoints, and 15 configurable IN endpoints.

The dedicated control interface can only be used for control transactions to endpoint 0 of Devices. These control transactions are used during enumeration or other control functions that communicate using endpoint 0 of Devices. This control endpoint shares the first 64 bytes of the USB controller’s FIFO RAM for IN and OUT transactions. The remaining IN and OUT interfaces can be configured to communicate with control, bulk, interrupt, or isochronous device endpoints.

Copyright © 2012–2019, Texas Instruments Incorporated

These USB interfaces can be used to simultaneously schedule as many as 15 independent OUT and 15 independent IN transactions to any endpoints on any device. The IN and OUT controls are paired in three sets of registers. However, they can be configured to communicate with different types of endpoints and different endpoints on devices. For example, the first pair of endpoint controls can be split so that the OUT portion is communicating with a device's bulk OUT endpoint 1, while the IN portion is communicating with a device's interrupt IN endpoint 2.

Before accessing any device, whether for point-to-point communications or for communications via a hub, the relevant USB Receive Functional Address Endpoint n (USBRXFUNCADDRn) or USB Transmit Functional Address Endpoint n (USBTXFUNCADDRn) registers must be set for each receive or transmit endpoint to record the address of the device being accessed.

The USB controller also supports connections to devices through a USB hub by providing a register that specifies the hub address and port of each USB transfer. The FIFO address and size are customizable and can be specified for each USB IN and OUT transfer. Customization includes allowing one FIFO per transaction, sharing a FIFO across transactions, and allowing for double-buffered FIFOs.

18.2.2.2 IN Transactions as a Host

IN transactions are handled in a similar manner to the way in which OUT transactions are handled when the USB controller is in device mode except that the transaction first must be initiated by setting the REQPKT bit in the USBCSRL0 register, indicating to the transaction scheduler that there is an active transaction on this endpoint. The transaction scheduler then sends an IN token to the target Device. When the packet is received and placed in the receive FIFO, the RXRDY bit in the USBCSRL0 register is set, and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, RXRDY must be cleared. The AUTOCL bit in the USBRXCSRHn register can be used to have RXRDY automatically cleared when a maximum-sized packet has been unloaded from the FIFO. The AUTORQ bit in USBRXCSRHn causes the REQPKT bit to be automatically set when the RXRDY bit is cleared. The AUTOCL and AUTORQ bits can be used with μDMA accesses to perform complete bulk transfers without main processor intervention. When the RXRDY bit is cleared, the controller sends an acknowledge to the Device. When there is a known number of packets to be transferred, the USB Request Packet Count in Block Transfer Endpoint n (USBRQPKTCOUNTn) register associated with the endpoint should be configured to the number of packets to be transferred. The USB controller decrements the value in the USBRQPKTCOUNTn register following each request. When the USBRQPKTCOUNTn value decrements to 0, the AUTORQ bit is cleared to prevent any further transactions being attempted. For cases where the size of the transfer is unknown, USBRQPKTCOUNTn should be cleared. AUTORQ then remains set until cleared by the reception of a short packet (that is, less than the MAXLOAD value in the USBRXMAXPn register) such as may occur at the end of a bulk transfer.

If the Device responds to a bulk or interrupt IN token with a NAK, the USB Host controller keeps retrying the transaction until any NAK Limit that has been set has been reached. If the target Device responds with a STALL, however, the USB Host controller does not retry the transaction but sets the STALLED bit in the USBCSRL0 register. If the target Device does not respond to the IN token within the required time, or the packet contained a CRC or bit-stuff error, the USB Host controller retries the transaction. If after three attempts the target Device has still not responded, the USB Host controller clears the REQPKT bit and sets the ERROR bit in the USBCSRL0 register.

18.2.2.2.1 OUT Transactions as a Host

OUT transactions are handled in a similar manner to the way in which IN transactions are handled when the USB controller is in device mode. The TXRDY bit in the USBTXCSRLn register must be set as each packet is loaded into the transmit FIFO. Again, setting the AUTOSET bit in the USBTXCSRHn register automatically sets TXRDY when a maximum-sized packet has been loaded into the FIFO. Furthermore, AUTOSET can be used with the μDMA controller to perform complete bulk transfers without software intervention.

Copyright © 2012–2019, Texas Instruments Incorporated

If the target device responds to the OUT token with a NAK, the USB Host controller keeps retrying the transaction until the NAK Limit that has been set has been reached. However, if the target device responds with a STALL, the USB controller does not retry the transaction but interrupts the main processor by setting the STALLED bit in the USBTXCSRLn register. If the target device does not respond to the OUT token within the required time, or the packet contained a CRC or bit-stuff error, the USB Host controller retries the transaction. If after three attempts the target device has still not responded, the USB controller flushes the FIFO and sets the ERROR bit in the USBTXCSRLn register.

18.2.2.3 Transaction Scheduling

Scheduling of transactions is handled automatically by the USB Host controller. The Host controller allows configuration of the endpoint communication scheduling based on the type of endpoint transaction. Interrupt transactions can be scheduled to occur in the range of every frame to every 255 frames in 1 frame increments. Bulk endpoints do not allow scheduling parameters, but do allow for a NAK timeout in the event an endpoint on a device is not responding. Isochronous endpoints can be scheduled from every frame to every 216 frames, in powers of 2.

The USB controller maintains a frame counter. If the target device is a full-speed device, the USB controller automatically sends an SOF packet at the start of each frame and increments the frame counter. If the target device is a low-speed device, a K state is transmitted on the bus to act as a keep-alive to stop the low-speed device from going into SUSPEND mode.

After the SOF packet has been transmitted, the USB Host controller cycles through all the configured endpoints looking for active transactions. An active transaction is defined as a receive endpoint for which the REQPKT bit is set or a transmit endpoint for which the TXRDY bit and/or the FIFONE bit is set.

An isochronous or interrupt transaction is started if the transaction is found on the first scheduler cycle of a frame and if the interval counter for that endpoint has counted down to zero. As a result, only one interrupt or isochronous transaction occurs per endpoint every n frames, where n is the interval set via the USB Host Transmit Interval Endpoint n (USBTXINTERVAL[*n*]) or USB Host Receive Interval Endpoint n (USBRXINTERVAL[*n*]) register for that endpoint.

An active bulk transaction starts immediately, provided sufficient time is left in the frame to complete the transaction before the next SOF packet is due. If the transaction must be retried (for example, because a NAK was received or the target device did not respond), then the transaction is not retried until the transaction scheduler has first checked all the other endpoints for active transactions. This process ensures that an endpoint that is sending a lot of NAKs does not block other transactions on the bus. The controller also allows the user to specify a limit to the length of time for NAKs to be received from a target device before the endpoint times out.

18.2.2.4 USB Hubs

The following setup requirements apply to the USB Host controller only if it is used with a USB hub. When a full- or low-speed device is connected to the USB controller via a USB 2.0 hub, details of the hub address and the hub port also must be recorded in the corresponding USB Receive Hub Address Endpoint n (USBRXHUBADDRn) and USB Receive Hub Port Endpoint n (USBRXHUBPORTn) or the USB Transmit Hub Address Endpoint n (USBTXHUBADDRn) and USB Transmit Hub Port Endpoint n (USBTXHUBPORTn) registers. In addition, the speed at which the device operates (full or low) must be recorded in the USB Type Endpoint 0 (USBTYPE0) (endpoint 0), USB Host Configure Transmit Type Endpoint n (USBTXTYPEn), or USB Host Configure Receive Type Endpoint n (USBRXTYPEn) registers for each endpoint that is accessed by the device.

For hub communications, the settings in these registers record the current allocation of the endpoints to the attached USB devices. To maximize the number of devices supported, the USB Host controller allows this allocation to be changed dynamically by simply updating the address and speed information recorded in these registers. Any changes in the allocation of endpoints to device functions must be made following the completion of any on-going transactions on the endpoints affected.

Copyright © 2012–2019, Texas Instruments Incorporated

18.2.2.5 Babble

The USB Host controller does not start a transaction until the bus has been inactive for at least the minimum inter-packet delay. The controller also does not start a transaction unless it can be finished before the end of the frame. If the bus is still active at the end of a frame, then the USB Host controller assumes that the target device to which it is connected has malfunctioned, and the USB controller suspends all transactions and generates a babble interrupt.

18.2.2.6 Host SUSPEND

If the SUSPEND bit in the USBPOWER register is set, the USB Host controller completes the current transaction then stops the transaction scheduler and frame counter. No further transactions are started and no SOF packets are generated.

To exit SUSPEND mode, set the RESUME bit and clear the SUSPEND bit. While the RESUME bit is set, the USB Host controller generates RESUME signaling on the bus. After 20 ms, the RESUME bit must be cleared, at which point the frame counter and transaction scheduler start. The Host supports the detection of a remote wake-up.

18.2.2.7 USB RESET

If the RESET bit in the USBPOWER register is set, the USB Host controller generates USB RESET signaling on the bus. The RESET bit must be set for at least 20 ms to ensure correct resetting of the target device. After the CPU has cleared the bit, the USB Host controller starts its frame counter and transaction scheduler.

18.2.2.8 Connect/Disconnect

A session is started by setting the SESSION bit in the USB device Control (USBDEVCTL) register, enabling the USB controller to wait for a device to be connected. When a device is detected, a connect interrupt is generated. The speed of the device that has been connected can be determined by reading the USBDEVCTL register where the FSDEV bit is set for a full-speed device, and the LSDEV bit is set for a low-speed device. The USB controller must generate a RESET to the device, and then the USB Host controller can begin device enumeration. If the device is disconnected while a session is in progress, a disconnect interrupt is generated.

18.2.3 OTG Mode

To conserve power, the USB On-The-Go (OTG) supplement allows VBUS to only be powered up when required and to be turned off when the bus is not in use. VBUS is always supplied by the A device on the bus. The USB OTG controller determines whether it is the A device or the B device by sampling the ID input from the PHY. This signal is pulled Low when an A-type plug is sensed (signifying that the USB OTG controller should act as the A device) but taken High when a B-type plug is sensed (signifying that the USB controller is a B device). Note that when switching between OTG A and OTG B, the USB controller retains all register contents.

18.2.3.1 Starting a Session

When the USB OTG controller is ready to start a session, the SESSION bit must be set in the USBDEVCTL register. The USB OTG controller then enables ID pin sensing. The ID input is either taken Low if an A-type connection is detected or High if a B-type connection is detected. The DEV bit in the USBDEVCTL register is also set to indicate whether the USB OTG controller has adopted the role of the A device or the B device. The USB OTG controller also provides an interrupt to indicate that ID pin sensing has completed and the mode value in the USBDEVCTL register is valid. This interrupt is enabled in the USBIDVIM register, and the status is checked in the USBIDVISC register. As soon as the USB controller has detected that it is on the A side of the cable, it must enable VBUS power within 100ms or the USB controller reverts to Device mode.

Copyright © 2012–2019, Texas Instruments Incorporated

If the USB OTG controller is the A device, then the USB OTG controller enters Host mode (the A device is always the default Host), turns on VBUS, and waits for VBUS to go above the VBUS Valid threshold, as indicated by the VBUS bit in the USBDEVCTL register going to 0x3. The USB OTG controller then waits for a peripheral to be connected. When a peripheral is detected, a Connect interrupt is signaled and either the FSDEV or LSDEV bit in the USBDEVCTL register is set, depending whether a full-speed or a low-speed peripheral is detected. The USB controller then issues a RESET to the connected Device. The SESSION bit in the USBDEVCTL register can be cleared to end a session. The USB OTG controller also automatically ends the session if babble is detected or if VBUS drops below session valid.

Note: The USB OTG controller may not remain in Host mode when connected to high-current devices. Some devices draw enough current to momentarily drop VBUS below the VBUS-valid level causing the controller to drop out of Host mode. The only way to get back into Host mode is to allow VBUS to go below the Session End level. In this situation, the device is causing VBUS to drop repeatedly and pull VBUS back low the next time VBUS is enabled.

In addition, the USB OTG controller may not remain in Host mode when a device is told that it can start using it's active configuration. At this point the device starts drawing more current and can also drop VBUS below VBUS valid.

If the USB OTG controller is the B device, then the USB OTG controller requests a session using the session request protocol defined in the USB On-The-Go supplement, that is, it first discharges VBUS. Then when VBUS has gone below the Session End threshold (VBUS bit in the USBDEVCTL register goes to 0x0) and the line state has been a single-ended zero for > 2 ms, the USB OTG controller pulses the data line, then pulses VBUS. At the end of the session, the SESSION bit is cleared either by the USB OTG controller or by the application software. The USB OTG controller then causes the PHY to switch out the pull-up resistor on D+, signaling the A device to end the session.

18.2.3.2 Detecting Activity

When the other device of the OTG setup wishes to start a session, it either raises VBUS above the Session Valid threshold if it is the A device, or if it is the B device, it pulses the data line then pulses VBUS. Depending on which of these actions happens, the USB controller can determine whether it is the A device or the B device in the current setup and act accordingly. If VBUS is raised above the Session Valid threshold, then the USB controller is the B device. The USB controller sets the SESSION bit in the USBDEVCTL register. When RESET signaling is detected on the bus, a RESET interrupt is signaled, which is interpreted as the start of a session.

The USB controller is in Device mode as the B device is the default mode. At the end of the session, the A device turns off the power to VBUS. When VBUS drops below the Session Valid threshold, the USB controller detects this drop and clears the SESSION bit to indicate that the session has ended, causing a disconnect interrupt to be signaled. If data line and VBUS pulsing is detected, then the USB controller is the A device. The controller generates a SESSION REQUEST interrupt to indicate that the B device is requesting a session. The SESSION bit in the USBDEVCTL register must be set to start a session.

18.2.3.3 Host Negotiation

When the USB controller is the A device, ID is Low, and the controller automatically enters Host mode when a session starts. When the USB controller is the B device, ID is High, and the controller automatically enters Device mode when a session starts. However, software can request that the USB controller become the Host by setting the HOSTREQ bit in the USBDEVCTL register. This bit can be set either at the same time as requesting a Session Start by setting the SESSION bit in the USBDEVCTL register or at any time after a session has started. When the USB controller next enters SUSPEND mode and if the HOSTREQ bit remains set, the controller enters Host mode and begins host negotiation (as specified in the USB On-The-Go supplement) by causing the PHY to disconnect the pull-up resistor on the D+ line, causing the A device to switch to Device mode and connect its own pull-up resistor. When the USB controller detects this, a Connect interrupt is generated and the RESET bit in the USBPOWER register is set to begin resetting the A device. The USB controller begins this reset sequence automatically to ensure that RESET is started as required within 1 ms of the A device connecting its pull-up resistor. The main processor should wait at least 20 ms, then clear the RESET bit and enumerate the A device.

Copyright © 2012–2019, Texas Instruments Incorporated

When the USB OTG controller B device has finished using the bus, the USB controller goes into SUSPEND mode by setting the SUSPEND bit in the USBPOWER register. The A device detects this and either terminates the session or reverts to Host mode. If the A device is USB OTG controller, it generates a disconnect interrupt.

18.2.4 DMA Operation

The USB peripheral provides an interface connected to the μDMA controller with separate channels for three 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 18-1. Remainder (MAXLOAD/4)

Value	Description
0	MAXLOAD = 64 bytes
1	MAXLOAD = 61 bytes
2	MAXLOAD = 62 bytes
3	MAXLOAD = 63 bytes

Table 18-2. Actual Bytes Read

Value	Description
0	MAXLOAD
1	MAXLOAD +3
2	MAXLOAD +2
3	MAXLOAD +1

Table 18-3. 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

Copyright © 2012–2019, Texas Instruments Incorporated

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 the *Micro Direct Memory Access* (μDMA) chapter for more details about programming the μDMA controller.

18.3 Initialization and Configuration

To use the USB Controller, the peripheral clock must be enabled via the RCGC2 register (see the *System Control* chapter). In addition, the clock to the appropriate GPIO module must be enabled via the RCGC2 register in the System Control module. Configure the PMCn fields in the GPIOPCTL register to assign the USB signals to the appropriate pins (see the GPIOs chapter).

The initial configuration in all cases requires that the processor enable the USB controller and USB controller's physical layer interface (PHY) before setting any registers. The next step is to enable the USB PLL so that the correct clocking is provided to the PHY. To ensure that voltage is not supplied to the bus incorrectly, the external power control signal, USB0EPEN, should be negated on start up by configuring the USB0EPEN and USB0PFLT pins to be controlled by the USB controller and not exhibit their default GPIO behavior.

Note: When used in OTG mode, USB0VBUS and USB0ID do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector's VBUS and ID signals. If the USB controller is used as either a dedicated Host or Device, the DEVMODOTG and DEVMOD bits in the USB General-Purpose Control and Status (USBGPCS) register can be used to connect the USB0VBUS and USB0ID inputs to fixed levels internally, freeing the pins for GPIO use. For proper self-powered Device operation, the VBUS value must still be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pull-up resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

18.3.1 Pin Configuration

When using the device controller portion of the USB controller in a system that also provides Host functionality, the power to VBUS must be disabled to allow the external Host controller to supply power. Usually, the USB0EPEN signal is used to control the external regulator and should be negated to avoid having two devices driving the USB0VBUS power pin on the USB connector.

When the USB controller is acting as a Host, it is in control of two signals that are attached to an external voltage supply that provides power to VBUS. The Host controller uses the USB0EPEN signal to enable or disable power to the USB0VBUS pin on the USB connector. An input pin, USB0PFLT, provides feedback when there has been a power fault on VBUS. The USB0PFLT signal can be configured to either automatically negate the USB0EPEN signal to disable power, and/or it can generate an interrupt to the interrupt controller to allow software to handle the power fault condition. The polarity and actions related to both USB0EPEN and USB0PFLT are fully configurable in the USB controller. The controller also provides interrupts on device insertion and removal to allow the Host controller code to respond to these external events.

18.3.2 Endpoint Configuration

To start communication in Host or device mode, the endpoint registers must first be configured. In Host mode, this configuration establishes a connection between an endpoint register and an endpoint on a device. In device mode, an endpoint must be configured before enumerating to the Host controller.

In both cases, the endpoint 0 configuration is limited because it is a fixed-function, fixed-FIFO-size endpoint. In Device and Host modes, the endpoint requires little setup but does require a software-based state machine to progress through the setup, data, and status phases of a standard control transaction. In device mode, the configuration of the remaining endpoints is done once before enumerating and then only changed if an alternate configuration is selected by the Host controller. In Host mode, the endpoints must be configured to operate as control, bulk, interrupt or isochronous mode. Once the type of endpoint is configured, a FIFO area must be assigned to each endpoint. In the case of bulk, control and interrupt endpoints, each has a maximum of 64 bytes per transaction. Isochronous endpoints can have packets with up to 1023 bytes per packet. In either mode, the maximum packet size for the given endpoint must be set prior to sending or receiving data.

Copyright © 2012–2019, Texas Instruments Incorporated

Configuring each endpoint's FIFO involves reserving a portion of the overall USB FIFO RAM to each endpoint. The total FIFO RAM available is 4 Kbytes with the first 64 bytes reserved for endpoint 0. The endpoint's FIFO must be at least as large as the maximum packet size. The FIFO can also be configured as a double-buffered FIFO so that interrupts occur at the end of each packet and allow filling the other half of the FIFO.

If operating as a device, the USB device controller's soft connect must be enabled when the device is ready to start communications, indicating to the host controller that the device is ready to start the enumeration process. If operating as a Host controller, the device soft connect must be disabled and power must be provided to VBUS via the USB0EPEN signal.

18.4 Register Map

Table 18-4 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 the *System Control* chapter).

Table 18-4. Universal Serial Bus (USB) Controller Register Map

Offset	Name	Type	Reset	Description	Section
0x000	USBFADDR[1]	R/W	0x00	USB Device Functional Address	Section 18.5.1
0x001	USBPOWER[1][2]	R/W	0x20	USB Power	Section 18.5.2
0x002	USBTXIS[1][2]	RO	0x0000	USB Transmit Interrupt Status	Section 18.5.3
0x004	USBRXIS[1][2]	RO	0x0000	USB Receive Interrupt Status	Section 18.5.4
0x006	USBTXIE[1][2]	R/W	0xFFFF	USB Transmit Interrupt Enable	Section 18.5.5
0x008	USBRXIE[1][2]	R/W	0xFFFE	USB Receive Interrupt Enable	Section 18.5.6
0x00A	USBIS[1][2]	RO	0x00	USB General Interrupt Status	Section 18.5.7
0x00B	USBIE[1][2]	R/W	0x06	USB Interrupt Enable	Section 18.5.8
0x00C	USBFRAME[1][2]	RO	0x0000	USB Frame Value	Section 18.5.9
0x00E	USBEPIDX[1][2]	R/W	0x00	USB Endpoint Index	Section 18.5.10
0x00F	USBTEST[1][2]	R/W	0x00	USB Test Mode	Section 18.5.11
0x020	USBFIFO0[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 0	Section 18.5.12
0x024	USBFIFO1[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 1	Section 18.5.12
0x028	USBFIFO2[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 2	Section 18.5.12
0x02C	USBFIFO3[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 3	Section 18.5.12
0x030	USBFIFO4[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 4	Section 18.5.12
0x034	USBFIFO5[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 5	Section 18.5.12
0x038	USBFIFO6[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 6	Section 18.5.12
0x03C	USBFIFO7[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 7	Section 18.5.12
0x040	USBFIFO8[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 8	Section 18.5.12
0x044	USBFIFO9[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 9	Section 18.5.12
0x048	USBFIFO10[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 10	Section 18.5.12
0x04C	USBFIFO11[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 11	Section 18.5.12
0x050	USBFIFO12[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 12	Section 18.5.12
0x054	USBFIFO13[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 13	Section 18.5.12
0x058	USBFIFO14[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 14	Section 18.5.12
0x05C	USBFIFO15[1][2]	R/W	0x0000.0000	USB FIFO Endpoint 15	Section 18.5.12
0x060	USBDEVCTL[2]	R/W	0x80	USB Device Control 53	Section 18.5.13
0x062	USBTXFIFOSZ[1][2]	R/W	0x00	USB Transmit Dynamic FIFO Sizing	Section 18.5.14
0x063	USBRXFIFOSZ[1][2]	R/W	0x00	USB Receive Dynamic FIFO Sizing	Section 18.5.15
0x064	USBTXFIFOADD[1][2]	R/W	0x0000	USB Transmit FIFO Start Address	Section 18.5.16
0x066	USBRXFIFOADD[1][2]	R/W	0x0000	USB Receive FIFO Start Address	Section 18.5.17
0x07A	USBCONTIM[1][2]	R/W	0x5C	USB Connect Timing	Section 18.5.18
0x07B	USBVPLEN[3]	R/W	0x3C	USB OTG VBUS Pulse Timing	Section 18.5.19

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-4. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Section
0x07D	USBFSEOF(1)(2)	R/W	0x77	USB Full-Speed Last Transaction to End of Frame Timing	Section 18.5.20
0x07E	USBLSEOF(1)(2)	R/W	0x72	USB Low-Speed Last Transaction to End of Frame Timing	Section 18.5.21
0x080	USBTXFUNCADDR0(2)	R/W	0x00	USB Transmit Functional Address Endpoint 0	Section 18.5.22
0x082	USBTXHUBADDR0(2)	R/W	0x00	USB Transmit Hub Address Endpoint 0	Section 18.5.23
0x083	USBTXHUBPORT0(2)	R/W	0x00	USB Transmit Hub Port Endpoint 0	Section 18.5.24
0x088	USBTXFUNCADDR1(2)	R/W	0x00	USB Transmit Functional Address Endpoint 1	Section 18.5.22
0x08A	USBTXHUBADDR1(2)	R/W	0x00	USB Transmit Hub Address Endpoint 1	Section 18.5.23
0x08B	USBTXHUBPORT1(2)	R/W	0x00	USB Transmit Hub Port Endpoint 1	Section 18.5.24
0x08C	USBRXFUNCADDR1(2)	R/W	0x00	USB Receive Functional Address Endpoint 1	Section 18.5.25
0x08E	USBRXHUBADDR1(2)	R/W	0x00	USB Receive Hub Address Endpoint 1	Section 18.5.26
0x08F	USBRXHUBPORT1(2)	R/W	0x00	USB Receive Hub Port Endpoint 1	Section 18.5.27
0x090	USBTXFUNCADDR2(2)	R/W	0x00	USB Transmit Functional Address Endpoint 2	Section 18.5.22
0x092	USBTXHUBADDR2(2)	R/W	0x00	USB Transmit Hub Address Endpoint 2	Section 18.5.23
0x093	USBTXHUBPORT2(2)	R/W	0x00	USB Transmit Hub Port Endpoint 2	Section 18.5.24
0x094	USBRXFUNCADDR2(2)	R/W	0x00	USB Receive Functional Address Endpoint 2	Section 18.5.25
0x096	USBRXHUBADDR2(2)	R/W	0x00	USB Receive Hub Address Endpoint 2	Section 18.5.26
0x097	USBRXHUBPORT2(2)	R/W	0x00	USB Receive Hub Port Endpoint 2	Section 18.5.27
0x098	USBTXFUNCADDR3(2)	R/W	0x00	USB Transmit Functional Address Endpoint 3	Section 18.5.22
0x09A	USBTXHUBADDR3(2)	R/W	0x00	USB Transmit Hub Address Endpoint 3	Section 18.5.23
0x09B	USBTXHUBPORT3(2)	R/W	0x00	USB Transmit Hub Port Endpoint 3	Section 18.5.24
0x09C	USBRXFUNCADDR3(2)	R/W	0x00	USB Receive Functional Address Endpoint 3	Section 18.5.25
0x09E	USBRXHUBADDR3(2)	R/W	0x00	USB Receive Hub Address Endpoint 3	Section 18.5.26
0x09F	USBRXHUBPORT3(2)	R/W	0x00	USB Receive Hub Port Endpoint 3	Section 18.5.27
0x0A0	USBTXFUNCADDR4(2)	R/W	0x00	USB Transmit Functional Address Endpoint 4	Section 18.5.22
0x0A2	USBTXHUBADDR4(2)	R/W	0x00	USB Transmit Hub Address Endpoint 4	Section 18.5.23
0x0A3	USBTXHUBPORT4(2)	R/W	0x00	USB Transmit Hub Port Endpoint 4	Section 18.5.24
0x0A4	USBRXFUNCADDR4(2)	R/W	0x00	USB Receive Functional Address Endpoint 4	Section 18.5.25
0x0A6	USBRXHUBADDR4(2)	R/W	0x00	USB Receive Hub Address Endpoint 4	Section 18.5.26
0x0A7	USBRXHUBPORT4(2)	R/W	0x00	USB Receive Hub Port Endpoint 4	Section 18.5.27
0x0A8	USBTXFUNCADDR5(2)	R/W	0x00	USB Transmit Functional Address Endpoint 5	Section 18.5.22
0x0AA	USBTXHUBADDR5(2)	R/W	0x00	USB Transmit Hub Address Endpoint 5	Section 18.5.23
0x0AB	USBTXHUBPORT5(2)	R/W	0x00	USB Transmit Hub Port Endpoint 5	Section 18.5.24
0x0AC	USBRXFUNCADDR5(2)	R/W	0x00	USB Receive Functional Address Endpoint 5	Section 18.5.25
0x0AE	USBRXHUBADDR5(2)	R/W	0x00	USB Receive Hub Address Endpoint 5	Section 18.5.26
0x0AF	USBRXHUBPORT5(2)	R/W	0x00	USB Receive Hub Port Endpoint 5	Section 18.5.27
0x0B0	USBTXFUNCADDR6(2)	R/W	0x00	USB Transmit Functional Address Endpoint 6	Section 18.5.22
0x0B2	USBTXHUBADDR6(2)	R/W	0x00	USB Transmit Hub Address Endpoint 6	Section 18.5.23
0x0B3	USBTXHUBPORT6(2)	R/W	0x00	USB Transmit Hub Port Endpoint 6	Section 18.5.24
0x0B4	USBRXFUNCADDR6(2)	R/W	0x00	USB Receive Functional Address Endpoint 6	Section 18.5.25
0x0B6	USBRXHUBADDR6(2)	R/W	0x00	USB Receive Hub Address Endpoint 6	Section 18.5.26
0x0B7	USBRXHUBPORT6(2)	R/W	0x00	USB Receive Hub Port Endpoint 6	Section 18.5.27
0x0B8	USBTXFUNCADDR7(2)	R/W	0x00	USB Transmit Functional Address Endpoint 7	Section 18.5.22
0x0BA	USBTXHUBADDR7(2)	R/W	0x00	USB Transmit Hub Address Endpoint 7	Section 18.5.23
0x0BB	USBTXHUBPORT7(2)	R/W	0x00	USB Transmit Hub Port Endpoint 7	Section 18.5.24
0x0BC	USBRXFUNCADDR7(2)	R/W	0x00	USB Receive Functional Address Endpoint 7	Section 18.5.25
0x0BE	USBRXHUBADDR7(2)	R/W	0x00	USB Receive Hub Address Endpoint 7	Section 18.5.26

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-4. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Section
0x0BF	USBRXHUBPORT7[2]	R/W	0x00	USB Receive Hub Port Endpoint 7	Section 18.5.27
0x0C0	USBTXFUNCADDR8[2]	R/W	0x00	USB Transmit Functional Address Endpoint 8	Section 18.5.22
0x0C2	USBTXHUBADDR8[2]	R/W	0x00	USB Transmit Hub Address Endpoint 8	Section 18.5.23
0x0C3	USBTXHUBPORT8[2]	R/W	0x00	USB Transmit Hub Port Endpoint 8	Section 18.5.24
0x0C4	USBRXFUNCADDR8[2]	R/W	0x00	USB Receive Functional Address Endpoint 8	Section 18.5.25
0x0C6	USBRXHUBADDR8[2]	R/W	0x00	USB Receive Hub Address Endpoint 8	Section 18.5.26
0x0C7	USBRXHUBPORT8[2]	R/W	0x00	USB Receive Hub Port Endpoint 8	Section 18.5.27
0x0C8	USBTXFUNCADDR9[2]	R/W	0x00	USB Transmit Functional Address Endpoint 9	Section 18.5.22
0x0CA	USBTXHUBADDR9[2]	R/W	0x00	USB Transmit Hub Address Endpoint 9	Section 18.5.23
0x0CB	USBTXHUBPORT9[2]	R/W	0x00	USB Transmit Hub Port Endpoint 9	Section 18.5.24
0x0CC	USBRXFUNCADDR9[2]	R/W	0x00	USB Receive Functional Address Endpoint 9	Section 18.5.25
0x0CE	USBRXHUBADDR9[2]	R/W	0x00	USB Receive Hub Address Endpoint 9	Section 18.5.26
0x0CF	USBRXHUBPORT9[2]	R/W	0x00	USB Receive Hub Port Endpoint 9	Section 18.5.27
0x0D0	USBTXFUNCADDR10[2]	R/W	0x00	USB Transmit Functional Address Endpoint 10	Section 18.5.22
0x0D2	USBTXHUBADDR10[2]	R/W	0x00	USB Transmit Hub Address Endpoint 10	Section 18.5.23
0x0D3	USBTXHUBPORT10[2]	R/W	0x00	USB Transmit Hub Port Endpoint 10	Section 18.5.24
0x0D4	USBRXFUNCADDR10[2]	R/W	0x00	USB Receive Functional Address Endpoint 10	Section 18.5.25
0x0D6	USBRXHUBADDR10[2]	R/W	0x00	USB Receive Hub Address Endpoint 10	Section 18.5.26
0x0D7	USBRXHUBPORT10[2]	R/W	0x00	USB Receive Hub Port Endpoint 10	Section 18.5.27
0x0D8	USBTXFUNCADDR11[2]	R/W	0x00	USB Transmit Functional Address Endpoint 11	Section 18.5.22
0x0DA	USBTXHUBADDR11[2]	R/W	0x00	USB Transmit Hub Address Endpoint 11	Section 18.5.23
0x0DB	USBTXHUBPORT11[2]	R/W	0x00	USB Transmit Hub Port Endpoint 11	Section 18.5.24
0x0DC	USBRXFUNCADDR11[2]	R/W	0x00	USB Receive Functional Address Endpoint 11	Section 18.5.25
0x0DE	USBRXHUBADDR11[2]	R/W	0x00	USB Receive Hub Address Endpoint 11	Section 18.5.26
0x0DF	USBRXHUBPORT11[2]	R/W	0x00	USB Receive Hub Port Endpoint 11	Section 18.5.27
0x0E0	USBTXFUNCADDR12[2]	R/W	0x00	USB Transmit Functional Address Endpoint 12	Section 18.5.22
0x0E2	USBTXHUBADDR12[2]	R/W	0x00	USB Transmit Hub Address Endpoint 12	Section 18.5.23
0x0E3	USBTXHUBPORT12[2]	R/W	0x00	USB Transmit Hub Port Endpoint 12	Section 18.5.24
0x0E4	USBRXFUNCADDR12[2]	R/W	0x00	USB Receive Functional Address Endpoint 12	Section 18.5.25
0x0E6	USBRXHUBADDR12[2]	R/W	0x00	USB Receive Hub Address Endpoint 12	Section 18.5.26
0x0E7	USBRXHUBPORT12[2]	R/W	0x00	USB Receive Hub Port Endpoint 12	Section 18.5.27
0x0E8	USBTXFUNCADDR13[2]	R/W	0x00	USB Transmit Functional Address Endpoint 13	Section 18.5.22
0x0EA	USBTXHUBADDR13[2]	R/W	0x00	USB Transmit Hub Address Endpoint 13	Section 18.5.23
0x0EB	USBTXHUBPORT13[2]	R/W	0x00	USB Transmit Hub Port Endpoint 13	Section 18.5.24
0x0EC	USBRXFUNCADDR13[2]	R/W	0x00	USB Receive Functional Address Endpoint 13	Section 18.5.25
0x0EE	USBRXHUBADDR13[2]	R/W	0x00	USB Receive Hub Address Endpoint 13	Section 18.5.26
0x0EF	USBRXHUBPORT13[2]	R/W	0x00	USB Receive Hub Port Endpoint 13	Section 18.5.27
0x0F0	USBTXFUNCADDR14[2]	R/W	0x00	USB Transmit Functional Address Endpoint 14	Section 18.5.22
0x0F2	USBTXHUBADDR14[2]	R/W	0x00	USB Transmit Hub Address Endpoint 14	Section 18.5.23
0x0F3	USBTXHUBPORT14[2]	R/W	0x00	USB Transmit Hub Port Endpoint 14	Section 18.5.24
0x0F4	USBRXFUNCADDR14[2]	R/W	0x00	USB Receive Functional Address Endpoint 14	Section 18.5.25
0x0F6	USBRXHUBADDR14[2]	R/W	0x00	USB Receive Hub Address Endpoint 14	Section 18.5.26
0x0F7	USBRXHUBPORT14[2]	R/W	0x00	USB Receive Hub Port Endpoint 14	Section 18.5.27
0x0F8	USBTXFUNCADDR15[2]	R/W	0x00	USB Transmit Functional Address Endpoint 15	Section 18.5.22
0x0FA	USBTXHUBADDR15[2]	R/W	0x00	USB Transmit Hub Address Endpoint 15	Section 18.5.23
0x0FB	USBTXHUBPORT15[2]	R/W	0x00	USB Transmit Hub Port Endpoint 15	Section 18.5.24
0x0FC	USBRXFUNCADDR15[2]	R/W	0x00	USB Receive Functional Address Endpoint 15	Section 18.5.25

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-4. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Section
0x0FE	USBRXHUBADDR15[2]	R/W	0x00	USB Receive Hub Address Endpoint 15	Section 18.5.26
0x0FF	USBRXHUBPORT15[2]	R/W	0x00	USB Receive Hub Port Endpoint 15	Section 18.5.27
0x102	USBCSRL0[1][2]	W1C	0x00	USB Control and Status Endpoint 0 Low	Section 18.5.29
0x103	USBCSRH0[1][2]	W1C	0x00	USB Control and Status Endpoint 0 High	Section 18.5.30
0x108	USBCOUNT0[1][2]	R/o	0x00	USB Receive Byte Count Endpoint 0	Section 18.5.31
0x10A	USBTYPE0[2]	R/W	0x00	USB Type Endpoint 0	Section 18.5.32
0x10B	USBNAKLMT[2]	R/W	0x00	USB NAK Limit	Section 18.5.33
0x110	USBTXMAXP1[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 1	Section 18.5.28
0x112	USBTXCSRL1[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 1 Low	Section 18.5.34
0x113	USBTXCSRH1[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 1 High	Section 18.5.35
0x114	USBRXMAXP1[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 1	Section 18.5.36
0x116	USBRXCSRL1[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 1 Low	Section 18.5.37
0x117	USBRXCSRH1[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 1 High	Section 18.5.38
0x118	USBRXCOUNT1[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 1	Section 18.5.39
0x11A	USBTXTYPE1[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 1	Section 18.5.40
0x11B	USBTXINTERVAL1[2]	R/W	0x00	USB Host Transmit Interval Endpoint 1	Section 18.5.41
0x11C	USBRXTYPE1[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 1	Section 18.5.42
0x11D	USBRXINTERVAL1[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 1	Section 18.5.43
0x120	USBTXMAXP2[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 2	Section 18.5.28
0x122	USBTXCSRL2[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 2 Low	Section 18.5.34
0x123	USBTXCSRH2[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 2 High	Section 18.5.35
0x124	USBRXMAXP2[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 2	Section 18.5.36
0x126	USBRXCSRL2[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 2 Low	Section 18.5.37
0x127	USBRXCSRH2[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 2 High	Section 18.5.38
0x128	USBRXCOUNT2[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 2	Section 18.5.39
0x12A	USBTXTYPE2[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 2	Section 18.5.40
0x12B	USBTXINTERVAL2[2]	R/W	0x00	USB Host Transmit Interval Endpoint 2	Section 18.5.41
0x12C	USBRXTYPE2[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 2	Section 18.5.42
0x12D	USBRXINTERVAL2[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 2	Section 18.5.43
0x130	USBTXMAXP3[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 3	Section 18.5.28
0x132	USBTXCSRL3[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 3 Low	Section 18.5.34
0x133	USBTXCSRH3[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 3 High	Section 18.5.35
0x134	USBRXMAXP3[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 3	Section 18.5.36
0x136	USBRXCSRL3[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 3 Low	Section 18.5.37
0x137	USBRXCSRH3[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 3 High	Section 18.5.38
0x138	USBRXCOUNT3[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 3	Section 18.5.39
0x13A	USBTXTYPE3[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 3	Section 18.5.40
0x13B	USBTXINTERVAL3[2]	R/W	0x00	USB Host Transmit Interval Endpoint 3	Section 18.5.41
0x13C	USBRXTYPE3[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 3	Section 18.5.42
0x13D	USBRXINTERVAL3[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 3	Section 18.5.43
0x140	USBTXMAXP4[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 4	Section 18.5.28
0x142	USBTXCSRL4[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 4 Low	Section 18.5.34
0x143	USBTXCSRH4[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 4 High	Section 18.5.35
0x144	USBRXMAXP4[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 4	Section 18.5.36
0x146	USBRXCSRL4[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 4 Low	Section 18.5.37
0x147	USBRXCSRH4[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 4 High	Section 18.5.38
0x148	USBRXCOUNT4[1][2]	R/W	0x0000	USB Receive Byte Count Endpoint 4	Section 18.5.39

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-4. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Section
0x14A	USBTXTYPE4[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 4	Section 18.5.40
0x14B	USBTXINTERVAL4[2]	R/W	0x00	USB Host Transmit Interval Endpoint 4	Section 18.5.41
0x14C	USBRXTYPE4[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 4	Section 18.5.42
0x14D	USBRXINTERVAL4[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 4	Section 18.5.43
0x150	USBTXMAXP5[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 5	Section 18.5.28
0x152	USBTXCSRL5[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 5 Low	Section 18.5.34
0x153	USBTXCSRH5[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 5 High	Section 18.5.35
0x154	USBRXMAXP5[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 5	Section 18.5.36
0x156	USBRXCSRL5[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 5 Low	Section 18.5.37
0x157	USBRXCSRH5[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 5 High	Section 18.5.38
0x158	USBRXCOUNT5[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 5	Section 18.5.39
0x15A	USBTXTYPE5[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 5	Section 18.5.40
0x15B	USBTXINTERVAL5[2]	R/W	0x00	USB Host Transmit Interval Endpoint 5	Section 18.5.41
0x15C	USBRXTYPE5[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 5	Section 18.5.42
0x15D	USBRXINTERVAL5[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 5	Section 18.5.43
0x160	USBTXMAXP6[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 6	Section 18.5.28
0x162	USBTXCSRL6[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 6 Low	Section 18.5.34
0x163	USBTXCSRH6[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 6 High	Section 18.5.35
0x164	USBRXMAXP6[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 6	Section 18.5.36
0x166	USBRXCSRL6[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 6 Low	Section 18.5.37
0x167	USBRXCSRH6[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 6 High	Section 18.5.38
0x168	USBRXCOUNT6[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 6	Section 18.5.39
0x16A	USBTXTYPE6[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 6	Section 18.5.40
0x16B	USBTXINTERVAL6[2]	R/W	0x00	USB Host Transmit Interval Endpoint 6	Section 18.5.41
0x16C	USBRXTYPE6[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 6	Section 18.5.42
0x16D	USBRXINTERVAL6[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 6	Section 18.5.43
0x170	USBTXMAXP7[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 7	Section 18.5.28
0x172	USBTXCSRL7[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 7 Low	Section 18.5.34
0x173	USBTXCSRH7[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 7 High	Section 18.5.35
0x174	USBRXMAXP7[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 7	Section 18.5.36
0x176	USBRXCSRL7[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 7 Low	Section 18.5.37
0x177	USBRXCSRH7[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 7 High	Section 18.5.38
0x178	USBRXCOUNT7[1][2]	RO	0x0000	USB Receive Byte Count Endpoint	Section 18.5.39
0x17A	USBTXTYPE7[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 7	Section 18.5.40
0x17B	USBTXINTERVAL7[2]	R/W	0x00	USB Host Transmit Interval Endpoint 7	Section 18.5.41
0x17C	USBRXTYPE7[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 7	Section 18.5.42
0x17D	USBRXINTERVAL7[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 7	Section 18.5.43
0x180	USBTXMAXP8[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 8	Section 18.5.28
0x182	USBTXCSRL8[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 8 Low	Section 18.5.34
0x183	USBTXCSRH8[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 8 High	Section 18.5.35
0x184	USBRXMAXP8[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 8	Section 18.5.36
0x186	USBRXCSRL8[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 8 Low	Section 18.5.37
0x187	USBRXCSRH8[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 8 High	Section 18.5.38
0x188	USBRXCOUNT8[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 8	Section 18.5.39
0x18A	USBTXTYPE8[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 8	Section 18.5.40
0x18B	USBTXINTERVAL8[2]	R/W	0x00	USB Host Transmit Interval Endpoint 8	Section 18.5.41
0x18C	USBRXTYPE8[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 8	Section 18.5.42

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-4. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Section
0x18D	USBRXINTERVAL8[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 8	Section 18.5.43
0x190	USBTXMAXP9[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 9	Section 18.5.28
0x192	USBTXCSRL9[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 9 Low	Section 18.5.34
0x193	USBTXCSRH9[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 9 High	Section 18.5.35
0x194	USBRXMAXP9[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 9	Section 18.5.36
0x196	USBRXCSRL9[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 9 Low	Section 18.5.37
0x197	USBRXCSRH9[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 9 High	Section 18.5.38
0x198	USBRXCOUNT9[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 9	Section 18.5.39
0x19A	USBTXTYPE9[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 9	Section 18.5.40
0x19B	USBTXINTERVAL9[2]	R/W	0x00	USB Host Transmit Interval Endpoint 9	Section 18.5.41
0x19C	USBRXTYPE9[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 9	Section 18.5.42
0x19D	USBRXINTERVAL9[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 9	Section 18.5.43
0x1A0	USBTXMAXP10[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 10	Section 18.5.28
0x1A2	USBTXCSRL10[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 10 Low	Section 18.5.34
0x1A3	USBTXCSRH10[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 10 High	Section 18.5.35
0x1A4	USBRXMAXP10[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 10	Section 18.5.36
0x1A6	USBRXCSRL10[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 10 Low	Section 18.5.37
0x1A7	USBRXCSRH10[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 10 High	Section 18.5.38
0x1A8	USBRXCOUNT10[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 10	Section 18.5.39
0x1AA	USBTXTYPE10[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 10	Section 18.5.40
0x1AB	USBTXINTERVAL10[2]	R/W	0x00	USB Host Transmit Interval Endpoint 10	Section 18.5.41
0x1AC	USBRXTYPE10[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 10	Section 18.5.42
0x1AD	USBRXINTERVAL10[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 10	Section 18.5.43
0x1B0	USBTXMAXP11[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 11	Section 18.5.28
0x1B2	USBTXCSRL11[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 11 Low	Section 18.5.34
0x1B3	USBTXCSRH11[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 11 High	Section 18.5.35
0x1B4	USBRXMAXP11[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 11	Section 18.5.36
0x1B6	USBRXCSRL11[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 11 Low	Section 18.5.37
0x1B7	USBRXCSRH11[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 11 High	Section 18.5.38
0x1B8	USBRXCOUNT11[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 11	Section 18.5.39
0x1BA	USBTXTYPE11[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 11	Section 18.5.40
0x1BB	USBTXINTERVAL11[2]	R/W	0x00	USB Host Transmit Interval Endpoint 11	Section 18.5.41
0x1BC	USBRXTYPE11[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 11	Section 18.5.42
0x1BD	USBRXINTERVAL11[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 11	Section 18.5.43
0x1C0	USBTXMAXP12[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 12	Section 18.5.28
0x1C2	USBTXCSRL12[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 12 Low	Section 18.5.34
0x1C3	USBTXCSRH12[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 12 High	Section 18.5.35
0x1C4	USBRXMAXP12[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 12	Section 18.5.36
0x1C6	USBRXCSRL12[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 12 Low	Section 18.5.37
0x1C7	USBRXCSRH12[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 12 High	Section 18.5.38
0x1C8	USBRXCOUNT12[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 12	Section 18.5.39
0x1CA	USBTXTYPE12[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 12	Section 18.5.40
0x1CB	USBTXINTERVAL12[2]	R/W	0x00	USB Host Transmit Interval Endpoint 12	Section 18.5.41
0x1CC	USBRXTYPE12[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 12	Section 18.5.42
0x1CD	USBRXINTERVAL12[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 12	Section 18.5.43
0x1D0	USBTXMAXP13[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 13	Section 18.5.28
0x1D2	USBTXCSRL13[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 13 Low	Section 18.5.34

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-4. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Section
0x1D3	USBTXCSRH13[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 13 High	Section 18.5.35
0x1D4	USBRXMAXP13[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 13	Section 18.5.36
0x1D6	USBRXCSRL13[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 13 Low	Section 18.5.37
0x1D7	USBRXCSRH13[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 13 High	Section 18.5.38
0x1D8	USBRXCOUNT13[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 13	Section 18.5.39
0x1DA	USBTXTYPE13[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 13	Section 18.5.40
0x1DB	USBTXINTERVAL13[2]	R/W	0x00	USB Host Transmit Interval Endpoint 13	Section 18.5.41
0x1DC	USBRXTYPE13[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 13	Section 18.5.42
0x1DD	USBRXINTERVAL13[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 13	Section 18.5.43
0x1E0	USBTXMAXP14[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 14	Section 18.5.28
0x1E2	USBTXCSRL14[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 14 Low	Section 18.5.34
0x1E3	USBTXCSRH14[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 14 High	Section 18.5.35
0x1E4	USBRXMAXP14[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 14	Section 18.5.36
0x1E6	USBRXCSRL14[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 14 Low	Section 18.5.37
0x1E7	USBRXCSRH14[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 14 High	Section 18.5.38
0x1E8	USBRXCOUNT14[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 14	Section 18.5.39
0x1EA	USBTXTYPE14[2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 14	Section 18.5.40
0x1EB	USBTXINTERVAL14[2]	R/W	0x00	USB Host Transmit Interval Endpoint 14	Section 18.5.41
0x1EC	USBRXTYPE14[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 14	Section 18.5.42
0x1ED	USBRXINTERVAL14[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 14	Section 18.5.43
0x1F0	USBTXMAXP15[1][2]	R/W	0x0000	USB Maximum Transmit Data Endpoint 15	Section 18.5.28
0x1F2	USBTXCSRL15[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 15 Low	Section 18.5.34
0x1F3	USBTXCSRH15[1][2]	R/W	0x00	USB Transmit Control and Status Endpoint 15 High	Section 18.5.35
0x1F4	USBRXMAXP15[1][2]	R/W	0x0000	USB Maximum Receive Data Endpoint 15	Section 18.5.36
0x1F6	USBRXCSRL15[1][2]	R/W	0x00	USB Receive Control and Status Endpoint 15 Low	Section 18.5.37
0x1F7	USBRXCSRH15 [1][2]	R/W	0x00	USB Receive Control and Status Endpoint 15 High	Section 18.5.38
0x1F8	USBRXCOUNT15[1][2]	RO	0x0000	USB Receive Byte Count Endpoint 15	Section 18.5.39
0x1FA	USBTXTYPE15 [2]	R/W	0x00	USB Host Transmit Configure Type Endpoint 15	Section 18.5.40
0x1FB	USBTXINTERVAL15[2]	R/W	0x00	USB Host Transmit Interval Endpoint 15	Section 18.5.41
0x1FC	USBRXTYPE15[2]	R/W	0x00	USB Host Configure Receive Type Endpoint 15	Section 18.5.42
0x1FD	USBRXINTERVAL15[2]	R/W	0x00	USB Host Receive Polling Interval Endpoint 15	Section 18.5.43
0x304	USBRQPKTCOUNT1[2]	R/W	0x0000 1	USB Request Packet Count in Block Transfer Endpoint 1	Section 18.5.44
0x308	USBRQPKTCOUNT2[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 2	Section 18.5.44
0x30C	USBRQPKTCOUNT3[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 3	Section 18.5.44
0x310	USBRQPKTCOUNT4[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 4	Section 18.5.44
0x314	USBRQPKTCOUNT5[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 5	Section 18.5.44
0x318	USBRQPKTCOUNT6[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 6	Section 18.5.44
0x31C	USBRQPKTCOUNT7[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 7	Section 18.5.44
0x320	USBRQPKTCOUNT8[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 8	Section 18.5.44
0x324	USBRQPKTCOUNT9[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 9	Section 18.5.44

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-4. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Section
0x328	USBRQPKTCOUNT10[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 10	Section 18.5.44
0x32C	USBRQPKTCOUNT11[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 11	Section 18.5.44
0x330	USBRQPKTCOUNT12[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 12	Section 18.5.44
0x334	USBRQPKTCOUNT13[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 13	Section 18.5.44
0x338	USBRQPKTCOUNT14[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 14	Section 18.5.44
0x33C	USBRQPKTCOUNT15[2]	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 15	Section 18.5.44
0x340	USBRXDPKTBUFDIS[1][2]	R/W	0x0000	USB Receive Double Packet Buffer Disable	Section 18.5.45
0x342	USBTXDPKTBUFDIS[1][2]	R/W	0x0000	USB Transmit Double Packet Buffer Disable	Section 18.5.46
0x400	USBEPC[1][2]	R/W	0x0000.0000	USB External Power Control	Section 18.5.47
0x404	USBEPCRIS[1][2]	RO	0x0000.0000	USB External Power Control Raw Interrupt Status	Section 18.5.48
0x408	USBEPCIM[1][2]	R/W	0x0000.0000	USB External Power Control Interrupt Mask	Section 18.5.49
0x40C	USBEPCISC[1][2]	R/W	0x0000.0000	USB External Power Control Interrupt Status and Clear	Section 18.5.50
0x410	USBDRRIS[1][2]	RO	0x0000.0000	USB Device RESUME Raw Interrupt Status	Section 18.5.51
0x414	USBDRIM[1][2]	R/W	0x0000.0000	USB Device RESUME Interrupt Mask	Section 18.5.52
0x418	USBDRISC[1][2]	W1C	0x0000.0000	USB Device RESUME Interrupt Status and Clear	Section 18.5.53
0x41C	USBGPCS[1][2]	R/W	0x0000.0000	USB General-Purpose Control and Status	Section 18.5.54
0x430	USBVDC[2]	R/W	0x0000.0000	USB VBUS Droop Control	Section 18.5.55
0x434	USBVDCRIS[2]	RO	0x0000.0000	USB VBUS Droop Control Raw Interrupt Status	Section 18.5.56
0x438	USBVDCIM[2]	R/W	0x0000.0000	USB VBUS Droop Control Interrupt Mask	Section 18.5.57
0x43C	USBVDCISC[2]	R/W	0x0000.0000	USB VBUS Droop Control Interrupt Status and Clear	Section 18.5.58
0x444	USBIDVRIS[3]	RO	0x0000.0000	USB ID Valid Detect Raw Interrupt Status	Section 18.5.59
0x448	USBIDVIM[3]	R/W	0x0000.0000	USB ID Valid Detect Interrupt Mask	Section 18.5.60
0x44C	USBIDVISC[3]	R/W1C	0x0000.0000	USB ID Valid Detect Interrupt Status and Clear	Section 18.5.61
0x450	USBDMASEL[1][2]	R/W	0x0033.2211	USB DMA Select	Section 18.5.62

(1) This register is used in OTG B or Device mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode.

(2) This register is used in OTG A or Host mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode. The USB controller is in OTG B or Device mode upon reset, so the reset values shown for these registers apply to the Device mode definition.

(3) This register is used for OTG-specific functions such as ID detection and negotiation. Once OTG negotiation is complete, then the USB controller registers are used according to their Host or Device mode meanings depending on whether the OTG negotiations made the USB controller OTG A (Host) or OTG B (Device).

Copyright © 2012–2019, Texas Instruments Incorporated

18.5 Register Descriptions

18.5.1 USB Device Functional Address Register (USBFADDR), offset 0x000

The USB function address 8-bit register (USBFADDR) contains the 7-bit address of the Device part of the transaction.

When the USB controller is being used in Device mode (the HOST bit in the USBDEVCTL register is clear), this register must be written with the address received through a SET_ADDRESS command, which is then used for decoding the function address in subsequent token packets.

Mode(s): OTG B or Device

For special considerations when writing this register, see the *Setting the Device Address* in Section 18.2.1.1.4.

USBFADDR is shown in Figure 18-2 and described in Table 18-5.

Figure 18-2. Function Address Register (USBFADDR)

7	6	0
Reserved	FUNCADDR	
R-0	R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 18-5. Function Address Register (USBFADDR) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	FUNCADDR	0-7Fh	Function Address of Device as received through SET_ADDRESS.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.2 USB Power Management Register (USBPOWER), offset 0x001

The power management 8-bit register (USBPOWER) is used for controlling SUSPEND and RESUME signaling, and some basic operational aspects of the USB controller.

Mode(s): OTG A or Host OTG B or Device

USBPOWER in OTG A/Host Mode is shown in Figure 18-3 and described in Table 18-6.

Figure 18-3. Power Management Register (USBPOWER) in OTG A/Host Mode

7 4	3	2	1	0
Reserved	RESET	RESUME	SUSPEND	PWRDNPHY
R-0	R/W-0	R/W-0	R/W-1S	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-6. Power Management Register (USBPOWER) in OTG A/Host Mode Field Descriptions

Bit	Field	Value	Description
7-4	Reserved	0	Reserved
3	RESET		RESET signaling.
		0	Ends RESET signaling on the bus.
		1	Enables RESET signaling on the bus.
2	RESUME		RESUME signaling. The bit should be cleared by software 20 ms after being set.
		0	Ends RESUME signaling on the bus.
		1	Enables RESUME signaling when the Device is in SUSPEND mode.
1	SUSPEND		SUSPEND mode
		0	No effect
		1	Enables SUSPEND mode.
0	PWRDNPHY		Power Down PHY
		0	No effect
		1	Powers down the internal USB PHY.

USBPOWER in OTG B/Device Mode is shown in Figure 18-4 and described in Table 18-7.

Figure 18-4. Power Management Register (USBPOWER) in OTG B/Device Mode

7	6	5 4	3	2	1	0
ISOUPDATE	SOFTCONN	Reserved	RESET	RESUME	SUSPEND	PWRDNPHY
R/W-0	R/W-0	R-0	R/W-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-7. Power Management Register (USBPOWER) in OTG B/Device Mode Field Descriptions

Bit	Field	Value	Description
7	ISOUPDATE		Isochronous Update
		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.
6	SOFTCONN		Soft Connect/Disconnect
		0	The USB D+/D- lines are tri-stated.
		1	The USB D+/D- lines are enabled.
5-4	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-7. Power Management Register (USBPOWER) in OTG B/Device Mode Field Descriptions (continued)

Bit	Field	Value	Description
3	RESET		RESET signaling
		0	Ends RESET signaling on the bus.
		1	Enables RESET signaling on the bus.
2	RESUME		RESUME signaling. The bit should be cleared by software 10 ms (a maximum of 15 ms) after being set.
		0	Ends RESUME signaling on the bus.
		1	Enables RESUME signaling when the Device is in SUSPEND mode.
1	SUSPEND		SUSPEND mode.
		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		Power Down PHY
		0	No effect
		1	Powers down the internal USB PHY.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.3 USB Transmit Interrupt Status Register (USBTXIS), offset 0x002

NOTE: Use caution when reading this register. Performing a read may change bit status.

The USB transmit interrupt status 16-bit read-only register (USBTXIS) indicates which interrupts are currently active for endpoint 0 and the transmit endpoints 1–15. The meaning of the EPn bits in this register is based on the mode of the device. The EP1 through EP15 bits always indicate that the USB controller is sending data; however, in Host mode, the bits refer to OUT endpoints; while in Device mode, the bits refer to IN endpoints. The EP0 bit is special in Host and Device modes and indicates that either a control IN or control OUT endpoint has generated an interrupt.

Mode(s): OTG A or Host OTG B or Device

USBTXIS is shown in Figure 18-5 and described in Table 18-8.

Figure 18-5. USB Transmit Interrupt Status Register (USBTXIS)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-8. USB Transmit Interrupt Status Register (USBTXIS) Field Descriptions

Bit	Field	Value	Description
15	EP15		TX Endpoint 15 Interrupt
		0	No interrupt
		1	The Endpoint 15 transmit interrupt is asserted.
14	EP14		TX Endpoint 14 Interrupt
		0	No interrupt
		1	The Endpoint 14 transmit interrupt is asserted.
13	EP13		TX Endpoint 13 Interrupt
		0	No interrupt
		1	The Endpoint 13 transmit interrupt is asserted.
12	EP12		TX Endpoint 12 Interrupt
		0	No interrupt
		1	The Endpoint 12 transmit interrupt is asserted.
11	EP11		TX Endpoint 11 Interrupt
		0	No interrupt
		1	The Endpoint 11 transmit interrupt is asserted.
10	EP10		TX Endpoint 10 Interrupt
		0	No interrupt
		1	The Endpoint 10 transmit interrupt is asserted.
9	EP9		TX Endpoint 9 Interrupt
		0	No interrupt
		1	The Endpoint 9 transmit interrupt is asserted.
8	EP8		TX Endpoint 8 Interrupt
		0	No interrupt
		1	The Endpoint 8 transmit interrupt is asserted.
7	EP7		TX Endpoint 7 Interrupt
		0	No interrupt
		1	The Endpoint 7 transmit interrupt is asserted.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-8. USB Transmit Interrupt Status Register (USBTXIS) Field Descriptions (continued)

Bit	Field	Value	Description
6	EP6		TX Endpoint 6 Interrupt
		0	No interrupt
		1	The Endpoint 6 transmit interrupt is asserted.
5	EP5		TX Endpoint 5 Interrupt
		0	No interrupt
		1	The Endpoint 5 transmit interrupt is asserted.
4	EP4		TX Endpoint 4 Interrupt
		0	No interrupt
		1	The Endpoint 4 transmit interrupt is asserted.
3	3P3		TX Endpoint 3 Interrupt
		0	No interrupt
		1	The Endpoint 3 transmit interrupt is asserted.
2	EP2		TX Endpoint 2 Interrupt
		0	No interrupt
		1	The Endpoint 2 transmit interrupt is asserted.
1	EP1		TX Endpoint 1 Interrupt
		0	No interrupt
		1	The Endpoint 1 transmit interrupt is asserted.
0	EP0		TX and RX Endpoint 0 Interrupt
		0	No interrupt
		1	The Endpoint 0 transmit and receive interrupt is asserted.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.4 USB Receive Interrupt Status Register (USBRXIS), offset 0x004

NOTE: Use caution when reading this register. Performing a read may change bit status.

The USB receive interrupt status 16-bit read-only register (USBRXIS) indicates which interrupts are currently active for receive endpoints 1–15.

Note: Bits relating to endpoints that have not been configured always return 0. All active interrupts are cleared when this register is read.

Mode(s): OTG A or Host OTG B or Device

USBRXIS is shown in Figure 18-6 and described in Table 18-9.

Figure 18-6. USB Receive Interrupt Status Register (USBRXIS)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	Rsvd
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-9. USB Receive Interrupt Status Register (USBRXIS) Field Descriptions

Bit	Field	Value	Description
15	EP15		RX Endpoint 15 Interrupt
		0	No interrupt
		1	The Endpoint 15 receive interrupt is asserted.
14	EP14		RX Endpoint 14 Interrupt
		0	No interrupt
		1	The Endpoint 14 receive interrupt is asserted.
13	EP13		RX Endpoint 13 Interrupt
		0	No interrupt
		1	The Endpoint 13 receive interrupt is asserted.
12	EP12		RX Endpoint 12 Interrupt
		0	No interrupt
		1	The Endpoint 12 receive interrupt is asserted.
11	EP11		RX Endpoint 11 Interrupt
		0	No interrupt
		1	The Endpoint 11 receive interrupt is asserted.
10	EP10		RX Endpoint 10 Interrupt
		0	No interrupt
		1	The Endpoint 10 receive interrupt is asserted.
9	EP9		RX Endpoint 9 Interrupt
		0	No interrupt
		1	The Endpoint 9 receive interrupt is asserted.
8	EP8		RX Endpoint 8 Interrupt
		0	No interrupt
		1	The Endpoint 8 receive interrupt is asserted.
7	EP7		RX Endpoint 7 Interrupt
		0	No interrupt
		1	The Endpoint 7 receive interrupt is asserted.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-9. USB Receive Interrupt Status Register (USBRXIS) Field Descriptions (continued)

Bit	Field	Value	Description
6	EP6		RX Endpoint 6 Interrupt
		0	No interrupt
		1	The Endpoint 6 receive interrupt is asserted.
5	EP5		RX Endpoint 5 Interrupt
		0	No interrupt
		1	The Endpoint 5 receive interrupt is asserted.
4	EP4		RX Endpoint 4 Interrupt
		0	No interrupt
		1	The Endpoint 4 receive interrupt is asserted.
3	3P3		RX Endpoint 3 Interrupt
		0	No interrupt
		1	The Endpoint 3 receive interrupt is asserted.
2	EP2		RX Endpoint 2 Interrupt
		0	No interrupt
		1	The Endpoint 2 receive interrupt is asserted.
1	EP1		RX Endpoint 1 Interrupt
		0	No interrupt
		1	The Endpoint 1 receive interrupt is asserted.
0	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.5 USB Transmit Interrupt Enable Register (USBTXIE), offset 0x006

The USB transmit interrupt enable 16-bit register (USBTXIE) 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.

Mode(s): OTG A or Host OTG B or Device

USBTXIS is shown in Figure 18-7 and described in Table 18-10.

Figure 18-7. USB Transmit Interrupt Status Enable Register (USBTXIE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-10. USB Transmit Interrupt Status Register (USBTXIE) Field Descriptions

Bit	Field	Value	Description
15	EP15		TX Endpoint 15 Interrupt Enable
		0	The EP15 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP15 bit in the USBTXIS register is set.
14	EP14		TX Endpoint 14 Interrupt Enable
		0	The EP14 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP14 bit in the USBTXIS register is set.
13	EP13		TX Endpoint 13 Interrupt Enable
		0	The EP13 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP13 bit in the USBTXIS register is set.
12	EP12		TX Endpoint 12 Interrupt Enable
		0	The EP12 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP12 bit in the USBTXIS register is set.
11	EP11		TX Endpoint 11 Interrupt Enable
		0	The EP11 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP11 bit in the USBTXIS register is set.
10	EP10		TX Endpoint 10 Interrupt Enable
		0	The EP10 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP10 bit in the USBTXIS register is set.
9	EP9		TX Endpoint 9 Interrupt Enable
		0	The EP9 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP9 bit in the USBTXIS register is set.
8	EP8		TX Endpoint 8 Interrupt Enable
		0	The EP8 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP8 bit in the USBTXIS register is set.
7	EP7		TX Endpoint 7 Interrupt Enable
		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		TX Endpoint 6 Interrupt Enable
		0	The EP6 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP6 bit in the USBTXIS register is set.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-10. USB Transmit Interrupt Status Register (USBTXIE) Field Descriptions (continued)

Bit	Field	Value	Description
5	EP5		TX Endpoint 5 Interrupt Enable
		0	The EP5 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP5 bit in the USBTXIS register is set.
4	EP4		TX Endpoint 4 Interrupt Enable
		0	The EP4 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP4 bit in the USBTXIS register is set.
3	3P3		TX Endpoint 3 Interrupt Enable
		0	The EP3 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP3 bit in the USBTXIS register is set.
2	EP2		TX Endpoint 2 Interrupt Enable
		0	The EP2 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP2 bit in the USBTXIS register is set.
1	EP1		TX Endpoint 1 Interrupt Enable
		0	The EP1 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP1 bit in the USBTXIS register is set.
0	EP0		TX and RX Endpoint 0 Interrupt Enable
		0	The EP0 transmit and receive interrupt is suppressed and not sent to the interupt controller.
		1	An interrupt is sent to the interrupt controller when the EP0 bit in the USBTXIS register is set.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.6 USB Receive Interrupt Enable Register (USBRXIE), offset 0x008

The USB receive interrupt enable 16-bit register (USBTXIE) 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.

Mode(s): OTG A or Host OTG B or Device

USBRXIE is shown in Figure 18-7 and described in Table 18-10.

Figure 18-8. USB Receive Interrupt Enable Register (USBRXIE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	Rsvd
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-11. USB Receive Interrupt Register (USBRXIE) Field Descriptions

Bit	Field	Value	Description
15	EP15		RX Endpoint 15 Interrupt Enable
		0	The EP15 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP15 bit in the USBRXIS register is set.
14	EP14		RX Endpoint 14 Interrupt Enable
		0	The EP14 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP14 bit in the USBRXIS register is set.
13	EP13		RX Endpoint 13 Interrupt Enable
		0	The EP13 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP13 bit in the USBRXIS register is set.
12	EP12		RX Endpoint 12 Interrupt Enable
		0	The EP12 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP12 bit in the USBRXIS register is set.
11	EP11		RX Endpoint 11 Interrupt Enable
		0	The EP11 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP11 bit in the USBRXIS register is set.
10	EP10		RX Endpoint 10 Interrupt Enable
		0	The EP10 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP10 bit in the USBRXIS register is set.
9	EP9		RX Endpoint 9 Interrupt Enable
		0	The EP9 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP9 bit in the USBRXIS register is set.
8	EP8		RX Endpoint 8 Interrupt Enable
		0	The EP8 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP8 bit in the USBRXIS register is set.
7	EP7		RX Endpoint 7 Interrupt Enable
		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		RX Endpoint 6 Interrupt Enable
		0	The EP6 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP6 bit in the USBRXIS register is set.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-11. USB Receive Interrupt Register (USBRXIE) Field Descriptions (continued)

Bit	Field	Value	Description
5	EP5		RX Endpoint 5 Interrupt Enable
		0	The EP5 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP5 bit in the USBRXIS register is set.
4	EP4		RX Endpoint 4 Interrupt Enable
		0	The EP4 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP4 bit in the USBRXIS register is set.
3	3P3		RX Endpoint 3 Interrupt Enable
		0	The EP3 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP3 bit in the USBRXIS register is set.
2	EP2		RX Endpoint 2 Interrupt Enable
		0	The EP2 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP2 bit in the USBRXIS register is set.
1	EP1		RX Endpoint 1 Interrupt Enable
		0	The EP1 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP1 bit in the USBRXIS register is set.
0	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.7 USB General Interrupt Status Register (USBIS), offset 0x00A

NOTE: Use caution when reading this register. Performing a read may change bit status.

The USB general interrupt status 8-bit read-only register (USBIS) indicates which USB interrupts are currently active. All active interrupts are cleared when this register is read.

Mode(s): OTG A or Host OTG B or Device

USBIS in OTG A/Host Mode is shown in Figure 18-9 and described in Table 18-12.

Figure 18-9. USB General Interrupt Status Register (USBIS) in OTG A/Host Mode

7	6	5	4	3	2	1	0
VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	Reserved
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-12. USB General Interrupt Status Register (USBIS) in OTG A/Host Mode Field Descriptions

Bit	Field	Value	Description
7	VBUSERR		VBUS Error
		0	No interrupt
		1	VBUS has dropped below the VBUS Valid threshold during a session.
6	SESREQ		Session Request
		0	No interrupt
		1	SESSION REQUEST signaling has been detected.
5	DISCON		Session Disconnect
		0	No interrupt
		1	A Device disconnect has been detected.
4	CONN		Session Connect
		0	No interrupt
		1	A Device connection has been detected.
3	SOF		Start of Frame
		0	No interrupt
		1	A new frame has started.
2	BABBLE		Babble Detected
		0	No interrupt
		1	Babble has been detected. This interrupt is active only after the first SOF has been sent.
1	RESUME		RESUME Signaling Detected. 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	No effect
		1	RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode.
0	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

USBIS in OTG B/Device Mode is shown in Figure 18-10 and described in Table 18-13.

Figure 18-10. USB General Interrupt Status Register (USBIS) in OTG B/Device Mode

7	6	5	4	3	2	1	0
Reserved		DISCON	Reserved	SOF	RESET	RESUME	SUSPEND
R-0		R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-13. USB General Interrupt Status Register (USBIS) in OTG B/Device Mode Field Descriptions

Bit	Field	Value	Description
7-6	Reserved	0	Reserved
5	DISCON		Session Disconnect
		0	No interrupt
		1	The device has been disconnected from the host.
4	Reserved	0	Reserved
3	SOF		Start of frame
		0	No interrupt
		1	A new frame has started.
2	RESET		RESET Signaling Detected
		0	No interrupt
		1	RESET signaling has been detected on the bus.
1	RESUME		RESUME Signaling Detected. 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	No interrupt
		1	RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode.
0	SUSPEND		SUSPEND Signaling Detected
		0	No interrupt
		1	SUSPEND signaling has been detected on the bus.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.8 USB Interrupt Enable Register (USBIE), offset 0x00B

NOTE: Use caution when reading this register. Performing a read may change bit status.

The USB interrupt enable 8-bit register (USBIE) provides interrupt enable bits for each of the interrupts in USBIS. At reset interrupts 1 and 2 are enabled in Device mode.

Mode(s): OTG A or Host OTG B or Device

USBIE in OTG A/Host Mode is shown in Figure 18-11 and described in Table 18-14.

Figure 18-11. USB Interrupt Enable Register (USBIE) in OTG A/Host Mode

7	6	5	4	3	2	1	0
VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	Reserved
R-W	R-W	R-W	R-W	R-W	R-W	R-W	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-14. USB Interrupt Enable Register (USBIE) in OTG A/Host Mode Field Descriptions

Bit	Field	Value	Description
7	VBUSERR		Enable VBUS Error Interrupt
		0	The VBUSERR interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the VBUSERR bit in the USBIS register is set.
6	SESREQ		Enable Session Request
		0	The SESREQ interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the SESREEQ bit in the USBIS register is set.
5	DISCON		Enable Disconnect Interrupt
		0	The DISCON interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set.
4	CONN		Enable Connect Interrupt
		0	The CONN interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the CONN bit in the USBIS register is set.
3	SOF		Start of Frame
		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	BABBLE		Babble Detected
		0	The BABBLE interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the BABBLE bit in the USBIS register is set.
1	RESUME		RESUME Signaling Detected. 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	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	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

USBIE in OTG B/Device Mode is shown in Figure 18-10 and described in Table 18-13.

Figure 18-12. USB Interrupt Enable Register (USBIE) in OTG B/Device Mode

7	6	5	4	3	2	1	0
Reserved		DISCON	Reserved	SOF	RESET	RESUME	SUSPEND
R-0		R/W-0	R-0	R/W-0	R/W-1	RW-1	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-15. USB Interrupt Enable Register (USBIE) in OTG B/Device Mode Field Descriptions

Bit	Field	Value	Description
7-6	Reserved	0	Reserved
5	DISCON		Enable Disconnect Interrupt
		0	The DISCON interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set.
4	Reserved	0	Reserved
3	SOF		Start of frame
		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		RESET Signaling Detected
		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		RESUME Signaling Detected. 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	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		SUSPEND Signaling Detected
		0	The SUSPEND interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.9 USB Frame Value Register (USBFRAME), offset 0x00C

The frame number 16-bit read-only register (USBFRAME) holds the last received frame number.

Mode(s): OTG A or Host OTG B or Device

USBFRAME is shown in Figure 18-13 and described in Table 18-16.

Figure 18-13. Frame Number Register (FRAME)

15	11	10	0
Reserved		FRAME	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 18-16. Frame Number Register (FRAME) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10-0	FRAME	0-7FFh	Last received frame number

18.5.10 USB Endpoint Index Register (USBEPIDX), offset 0x00E

Each endpoint's buffer can be accessed by configuring a FIFO size and starting address. The endpoint index 16-bit register (USBEPIDX) is used with the USBTXFIFOSZ, USBRXFIFOSZ, USBTXFIFOADD, and USBRXFIFOADD registers.

Mode(s): OTG A or Host OTG B or Device

USBEPIDX is shown in Figure 18-14 and described in Table 18-17.

Figure 18-14. USB Endpoint Index Register (USBEPIDX)

7	4	3	0
Reserved		EPIDX	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-17. USB Endpoint Index Register (USBEPIDX) Field Descriptions

Bit	Field	Value	Description
7-4	Reserved	0	Reserved
3-0	EPIDX	0-4h	Endpoint Index. This bit field configures which endpoint is accessed when reading or writing to one of the USB controller's indexed registers. A value of 0x0 corresponds to Endpoint 0 and a value of 0xF corresponds to Endpoint 15.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.11 USB Test Mode Register (USBTEST), offset 0x00F

The USB test mode 8-bit register (USBTEST) 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.

Mode(s): OTG A or Host OTG B or Device

USBTEST in OTG A/Host Mode is shown in Figure 18-15 and described in Table 18-18.

Figure 18-15. USB Test Mode Register (USBTEST) in OTG A/Host Mode

7	6	5	4	0
FORCEH	FIFOACC	FORCEFS	Reserved	
R/W-0	R/W1S-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; W = Write only; -*n* = value after reset

Table 18-18. USB Test Mode Register (USBTEST) in OTG A/Host Mode Field Descriptions

Bit	Field	Value	Description
7	FORCEH		Force Host Mode. While in this mode, status of the bus connection may be read using the DEV bit of the USBDEVCTL register. The operating speed is determined from the FORCEFS bit.
		0	No effect
		1	Forces the USB controller to enter Host mode when the SESSION bit is set, regardless of whether the USB controller is connected to any peripheral. The state of the USB0DP and USB0DM signals is ignored. The USB controller then remains in Host mode until the SESSION bit is cleared, even if a Device is disconnected. If the FORCEH bit remains set, the USB controller re-enters Host mode the next time the SESSION bit is set.
6	FIFOACC		FIFO Access
		0	No effect
		1	Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO.
5	FORCEFS		Force Full-Speed Mode
		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	0	Reserved

USBTEST in OTG B/Device Mode is shown in Figure 18-16 and described in Table 18-19.

Figure 18-16. USB Test Mode Register (USBTEST) in OTG B/Device Mode

7	6	5	4	0
Reserved	FIFOACC	FORCEFS	Reserved	
R-0	R/W1S-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; W = Write only; -*n* = value after reset

Table 18-19. USB Test Mode Register (USBTEST) in OTG B/Device Mode Field Descriptions

Bit	Field	Value	Description
7	Reserved		Force Host Mode. While in this mode, status of the bus connection may be read using the DEV bit of the USBDEVCTL register. The operating speed is determined from the FORCEFS bit.
6	FIFOACC		FIFO Access
		0	No effect
		1	Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-19. USB Test Mode Register (USBTEST) in OTG B/Device Mode Field Descriptions (continued)

Bit	Field	Value	Description
5	FORCEFS		Force Full-Speed Mode
		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	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.12 USB FIFO Endpoint n Register (USBFIFO[0]-USBFIFO[15])

NOTE: Use caution when reading these registers. Performing a read may change bit status.

The USB FIFO endpoint *n* 32-bit registers (USBFIFO[*n*]) 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 can 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 *Single-Packet Buffering* in Section 18.2.1.1.2). Burst writing of multiple packets is not supported as flags must be set after each packet is written.

Following a STALL response or a transmit error on endpoint 1–15, the associated FIFO is completely flushed.

For the specific offset for each FIFO register see Table 18-4.

Mode(s): OTG A or Host OTG B or Device

USBFIFO0-15 are shown in Figure 18-17 and described in Table 18-20.

Figure 18-17. USB FIFO Endpoint *n* Register (USBFIFO[*n*])

31	0
EPDATA	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-20. USB FIFO Endpoint *n* Register (USBFIFO[*n*]) Field Descriptions

Bit	Field	Reset	Description
31-0	EPDATA	0x0000.0000	Endpoint Data. Writing to this register loads the data into the Transmit FIFO and reading unloads data from the Receive FIFO.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.13 USB Device Control Register (USBDEVCTL), offset 0x060

The USB device control 8-bit register (USBDEVCTL) is used for controlling and monitoring the USB VBUS line. If the PHY is suspended, no PHY clock is received and the VBUS is not sampled. In addition, in Host mode, USBDEVCTL provides the status information for the current operating mode (Host or Device) of the USB controller. If the USB controller is in Host mode, this register also indicates if a full- or low-speed Device has been connected.

Mode(s): OTG A or Host

USBDEVCTL is shown in Figure 18-18 and described in Table 18-21.

Figure 18-18. USB Device Control Register (USBDEVCTL)

7	6	5	4	3	2	1	0
DEV	FSDEV	LSDEV	VBUS		HOSTMODE	HOSTREQ	SESSION
R-1	R-0	R-0	R-0		R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-21. USB Device Control Register (USBDEVCTL) Field Descriptions

Bit	Field	Value	Description
7	DEV		Device mode
		0	The USB controller is operating on the OTG A side of the cable.
		1	The USB controller is operating on the OTG B side of the cable.
			Only valid while a session is in progress.
6	FSDEV		Full-Speed Device Detected
		0	A full-speed Device has not been detected on the port.
		1	A full-speed Device has been detected on the port.
5	LSDEV		Low-Speed Device Detected
		0	A low-speed Device has not been detected on the port.
		1	A low-speed Device has been detected on the port.
4-3	VBUS	0-3h	These read-only bits encode the current VBus level as follows:
		0	Below Session End. VBUS is detected as under 0.5 V.
		1h	Above Session End, below AValid. VBUS is detected as above 0.5 V and under 1.5 V.
		2h	Above AValid, below VBusValid. VBUS is detected as above 1.5 V and below 4.75 V.
		3h	Above VBusValid. VBUS is detected as above 4.75 V.
2	HOSTMODE		This read-only bit is set when the USB controller is acting as a Host.
		0	The USB controller is acting as a Device.
		1	The USB controller is acting as a Host.
			Only valid while a session is in progress.
1	HOSTREQ		When set, the USB controller will initiate the Host Negotiation when Suspend mode is entered. It is cleared when Host Negotiation is completed.
		0	No effect
		1	Initiates the Host Negotiation when SUSPENDmode is entered.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-21. USB Device Control Register (USBDEVCTL) Field Descriptions (continued)

Bit	Field	Value	Description
0	SESSION		Session Start/End When operating as an OTG A device:
		0	When cleared by software, this bit ends a session.
		1	When set by software, this bit starts a session.
			When operating as an OTG A device:
		0	The USB controller has ended a session. When the USB controller is in SUSPEND mode, this bit may be cleared by software to perform a software disconnect.
		1	The USB controller has started a session. When set by software, the Session Request Protocol is initiated.
			Clearing this bit when the USB controller is not suspended results in undefined behavior.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.14 USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ), offset 0x062

The USB transmit dynamic FIFO sizing 8-bit register (USBTXFIFOSZ) allows the selected TX endpoint FIFOs to be dynamically sized. USBEPIDX is used to configure each transmit endpoint's FIFO size.

Mode(s): OTG A or Host OTG B or Device

USBTXFIFOSZ is shown in Figure 18-19 and described in Table 18-22.

Figure 18-19. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ)

7	5	4	3	0
Reserved		DPB	SZ	
R-0		R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-22. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved	0	Reserved
4	DPB		Double Packet Buffering Support
		0	Single packet buffering is supported.
		1	Double packet buffering is enabled.
3-0	SZ		Maximum packet size to be allowed. If DPB = 0, the FIFO also is this size; if DPB = 1, the FIFO is twice this size. Packet size in bytes:
		0h	8
		1h	16
		2h	32
		3h	64
		4h	128
		5h	256
		6h	512
		7h	1024
		8h	2048
		9-Fh	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.15 USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ), offset 0x063

The USB receive dynamic FIFO sizing 8-bit register (USBRXFIFOSZ) allows the selected RX endpoint FIFOs to be dynamically sized.

Mode(s): OTG A or Host OTG B or Device

USBRXFIFOSZ is shown in Figure 18-20 and described in Table 18-23.

Figure 18-20. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ)

7	5	4	3	0
Reserved		DPB	SZ	
R-0		R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-23. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved	0	Reserved
4	DPB		Double Packet Buffering Support
		0	Single packet buffering is supported.
		1	Double packet buffering is enabled.
3-0	SZ		Maximum packet size to be allowed. If DPB = 0, the FIFO also is this size; if DPB = 1, the FIFO is twice this size. Packet size in bytes:
		0h	8
		1h	16
		2h	32
		3h	64
		4h	128
		5h	256
		6h	512
		7h	1024
		8h	2048
		9-Fh	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.16 USB Transmit FIFO Start Address Register (USBTXFIFOADD), offset 0x064

The USB transmit FIFO start address 16-bit register (USBTXFIFOADD) controls the start address of the selected transmit endpoint FIFOs.

Mode(s): OTG A or Host OTG B or Device

USBTXFIFOADDR is shown in Figure 18-21 and described in Table 18-24.

Figure 18-21. USB Transmit FIFO Start Address Register (USBTXFIFOADDR])

15	9	8	0
Reserved		ADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-24. USB Transmit FIFO Start Address Register (USBTXFIFOADDR) Field Descriptions

Bit	Field	Value	Description
15-9	Reserved	0	Reserved
8-0	ADDR		Start Address of the endpoint FIFO in units of 8 bytes.
		0h	0
		1h	8
		2h	16
		3h	24
		4h	32
		5h	40
		6h	48
		7h	56
		8h	64
		..	..
		1FFh	4095

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.17 USB Receive FIFO Start Address Register (USBRXFIFOADD), offset 0x066

The USB receive FIFO start address 16-bit register (USBRXFIFOADD) controls the start address of the selected receive endpoint FIFOs.

Mode(s): OTG A or Host OTG B or Device

USBRXFIFOADDR is shown in Figure 18-22 and described in Table 18-25.

Figure 18-22. USB Receive FIFO Start Address Register (USBRXFIFOADDR)

15	9	8	0
Reserved		ADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-25. USB Receive FIFO Start Address Register (USBRXFIFOADDR) Field Descriptions

Bit	Field	Value	Description
15-9	Reserved	0	Reserved
8-0	ADDR		Start Address of the endpoint FIFO in units of 8 bytes.
		0h	0
		1h	8
		2h	16
		3h	24
		4h	32
		5h	40
		6h	48
		7h	56
		8h	64
		..	..
		1FFh	4095

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.18 USB Connect Timing Register (USBCONTIM), offset 0x07A

The USB connect timing 8-bit configuration register (USBCONTIM) specifies connection and negotiation delays.

Mode(s): OTG A or Host OTG B or Device

USBCONTIM is shown in Figure 18-23 and described in Table 18-26.

Figure 18-23. USB Connect Timing Register (USBCONTIM)

7	6	5	4	3	2	1	0
WTCON				WTID			
R/W-0	R/W-1	R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-26. USB Connect Timing Register (USBCONTIM) Field Descriptions

Bit	Field	Value	Description
7-4	WTCON	5h	The connect wait 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	Ch	The wait ID 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.

18.5.19 USB OTG VBUS Pulse Timing Register (USBVPLEN), offset 0x07B

The USB OTG VBUS pulse timing 8-bit configuration register (USBVPLEN) specifies the duration of the VBUS pulsing charge.

Mode(s): OTG-specific

USBVPLEN is shown in Figure 18-24 and described in Table 18-27.

Figure 18-24. USB OTG VBUS Pulse Timing Register (USBVPLEN)

7	6	5	4	3	2	1	0
VPLEN							
R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-27. USB OTG VBUS Pulse Timing Register (USBVPLEN) Field Descriptions

Bit	Field	Reset	Description
7-0	VPLEN	3h	The VBUS pulse length field configures the duration of the VBUS pulsing charge in units of 546.1 µs. The default corresponds to 32.77 ms.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.20 USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF), offset 0x07D

USB full-speed last transaction to end of frame timing 8-bit configuration register (USBFSEOF) specifies the minimum time gap allowed between the start of the last transaction and the EOF for full-speed transactions.

Mode(s): OTG A or Host OTG B or Device

USBFSEOF is shown in Figure 18-25 and described in Table 18-28.

Figure 18-25. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF)

7							0
FSEOFG							
R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-28. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF) Field Descriptions

Bit	Field	Reset	Description
7-0	FSEOFG	77h	The full-speed end-of-frame gap 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.

18.5.21 USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF), offset 0x07E

The USB low-speed last transaction to end of frame timing 8-bit configuration register (USBLSEOF) specifies the minimum time gap that is to be allowed between the start of the last transaction and the EOF for low-speed transactions.

Mode(s): OTG A or Host OTG B or Device

USBLSEOF is shown in Figure 18-26 and described in Table 18-29.

Figure 18-26. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF)

7							0
LSEOFG							
R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0	R/W-1	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-29. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF) Field Descriptions

Bit	Field	Reset	Description
7-0	LSEOFG	72h	The low-speed end-of-frame gap 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.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.22 USB Transmit Functional Address Endpoint n Registers (USBTXFUNCADDR[0]-USBTXFUNCADDR[15])

The transmit functional address endpoint *n* 8-bit registers (USBTXFUNCADDR[*n*]) record the address of the target function to be accessed through the associated endpoint (EP*n*). USBTXFUNCADDRx must be defined for each transmit endpoint that is used.

Note: USBTXFUNCADDR0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBTXFUNCADDR[*n*] registers are shown in Figure 18-27 and described in Table 18-30.

Figure 18-27. USB Transmit Functional Address Endpoint *n* Registers (USBTXFUNCADDR[*n*])

7	6	0
Reserved	ADDR	
R-0	R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-30. USB Transmit Functional Address Endpoint *n* Registers (USBTXFUNCADDR[*n*]) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	ADDR	0	Device Address specifies the USB bus address for the target Device.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.23 USB Transmit Hub Address Endpoint n Registers (USBTXHUBADDR[0]-USBTXHUBADDR[15])

The transmit hub address endpoint *n* 8-bit read/write registers (USBTXHUBADDR[*n*]), like USBTXHUBPORT[*n*], must be written only when a USB device is connected to transmit endpoint EP*n* via a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBTXHUBADDR[*n*] registers are shown in Figure 18-27 and described in Table 18-30.

Figure 18-28. USB Transmit Hub Address Endpoint *n* Registers (USBTXHUBADDR[*n*])

7	6	0
Reserved	ADDR	
R-0	R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-31. USB Transmit Hub Address Endpoint *n* Registers(USBTXHUBADDR[*n*]) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	ADDR	0	Device Address specifies the USB bus address for the target Device.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.24 USB Transmit Hub Port Endpoint n Registers (USBTXHUBPORT[0]-USBTXHUBPORT[15])

The transmit hub port endpoint *n* 8-bit read/write registers (USBTXHUBPORT[*n*]), like USBTXHUBADDR[*n*], must be written only when a full- or low-speed Device is connected to transmit endpoint EP*n* via a USB 2.0 hub. This register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: USBTXHUBPORT0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBTXHUBPORT*n* registers are shown in Figure 18-29 and described in Table 18-32.

Figure 18-29. USB Transmit Hub Port Endpoint *n* Registers (USBTXHUBPORT[*n*])

7	6	0
Reserved	PORT	
R-0	R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-32. USB Transmit Hub Port Endpoint *n* Registers(USBTXHUBPORT[*n*]) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	PORT	0	Hub Port specifies the USB hub port number.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.25 USB Receive Functional Address Endpoint n Registers (USBRXFUNCADDR[1]-USBRXFUNCADDR[15])

The recieve functional address endpoint *n* 8-bit read/write registers (USBRXFUNCADDR[*n*]) record the address of the target function to be accessed through the associated endpoint (EP*n*). USBRXFUNCADDRx must be defined for each receive endpoint that is used.

Note: USBTXFUNCADDR0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBRXFUNCADDR[*n*] registers are shown in Figure 18-30 and described in Table 18-33.

Figure 18-30. USB Receive Functional Address Endpoint *n* Registers (USBFIFO[*n*])

7	6	0
Reserved	ADDR	
R-0	R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-33. USB Recieve Functional Address Endpoint *n* Registers(USBFIFO[*n*]) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	ADDR	0	Device Address specifies the USB bus address for the target Device.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.26 USB Receive Hub Address Endpoint n Registers (USBRXHUBADDR[1]-USBRXHUBADDR[15])

The receive hub address endpoint *n* 8-bit read/write registers (USBRXHUBADDR[*n*]), like [*n*], must be written only when a full- or low-speed Device is connected to receive endpoint EP*n* via a USB 2.0 hub. Each register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBRXHUBADDR[*n*] registers are shown in Figure 18-31 and described in Table 18-34.

Figure 18-31. USB Receive Hub Address Endpoint *n* Registers (USBRXHUBADDR[*n*])

7	6	0
MULTTRAN	ADDR	
R/w-0	R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-34. USB Receive Hub Address Endpoint *n* Registers(USBRXHUBADDR[*n*]) Field Descriptions

Bit	Field	Value	Description
7	MULTTRAN		Multiple Translators
		0	Clear to indicate that the hub has a single transaction translator.
		1	Set to indicate that the hub has multiple transaction translators.
6-0	ADDR	0	Device Address specifies the USB bus address for the target Device.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.27 USB Receive Hub Port Endpoint n Registers (USBRXHUBPORT[1]-USBRXHUBPORT[15])

The receive hub port endpoint *n* 8-bit read/write registers (USBRXHUBPORT[*n*]), like USBRXHUBADDR[*n*], must be written only when a full- or low-speed Device is connected to receive endpoint EP*n* via a USB 2.0 hub. Each register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: USBTXHUBPORT0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBRXHUBPORT*n* registers are shown in Figure 18-32 and described in Table 18-35.

Figure 18-32. USB Transmit Hub Port Endpoint *n* Registers (USBRXHUBPORT[*n*])

7	6	0
Reserved	PORT	
R-0	R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-35. USB Transmit Hub Port Endpoint *n* Registers(USBRXHUBPORT[*n*]) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	PORT	0	Hub Port specifies the USB hub port number.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.28 USB Maximum Transmit Data Endpoint n Registers (USBTXMAXP[1]-USBTXMAXP[15])

The USB maximum transmit data endpoint *n* 16-bit registers (USBTXMAXP[*n*]) define the maximum amount of data that can be transferred through the selected 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: USBTXMAXP[*n*] must be set to an even number of bytes for proper interrupt generation in μDMA Basic Mode.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host OTG B or Device

The USBTXMAXP[*n*] registers are shown in Figure 18-33 and described in Table 18-36.

Figure 18-33. USB Maximum Transmit Data Endpoint *n* Registers (USBTXMAXP[*n*])

15	11	10	0
Reserved		MAXLOAD	
R-0		R/W-000	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-36. USB Maximum Transmit Data Endpoint *n* Registers(USBTXMAXP[*n*]) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10-0	MAXLOAD		Maximum Payload specifies the maximum payload in bytes per transaction.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.29 USB Control and Status Endpoint 0 Low Register (USBCSRL0), offset 0x102

The USB control and status endpoint 0 low 8-bit register (USBCSRL0) provides control and status bits for endpoint 0.

Mode(s): OTG A or Host OTG B or Device

USBCSRL0 in OTG A/Host mode is shown in Figure 18-34 and described in Table 18-37.

Figure 18-34. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in OTG A/Host Mode

7	6	5	4	3	2	1	0
NAKTO	STATUS	REQPKT	ERROR	SETUP	STALLED	TXRDY	RXRDY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 18-37. USB Control and Status Endpoint 0 Low Register(USBCSRL0) in OTG A/Host Mode Field Descriptions

Bit	Field	Value	Description
7	NAKTO		NAK Timeout. Software must clear this bit to allow the endpoint to continue.
		0	No timeout
		1	Indicates that endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the USBNAKLMT register.
6	STATUS		Status Packet. Setting this bit ensures that the DT bit is set in the USBCSRH0 register so that a DATA1 packet is used for the STATUS stage transaction.
		0	No transaction
		1	Initiates a STATUS stage transaction. This bit must be set at the same time as the TXRDY or REQPKT bit is set.
			This bit is automatically cleared when the STATUS stage is over.
5	REQPKT		Request Packet. This bit is cleared when the RXRDY bit is set.
		0	No request
		1	Requests an IN transaction.
4	ERROR		Error. Software must clear this bit.
		0	No error
		1	Three attempts have been made to perform a transaction with no response from the peripheral. The EP0 bit in the USBTXIS register is also set in this situation.
3	SETUP		Setup Packet. Setting this bit always clears the DT bit in the USBCSRH0 register to send a DATA0 packet.
		0	Sends an OUT token.
		1	Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.
2	STALLED		Endpoint Stalled. Software must clear this bit.
		0	No handshake has been received.
		1	A STALL handshake has been received.
1	TXRDY		Transmit Packet Ready. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent.
		0	No transmit packet is ready.
		1	Software sets this bit after loading a data packet into the TX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.
0	RXRDY		Receive Packet Ready. Software must clear this bit after the packet has been read from the FIFO to acknowledge that the data has been read from the FIFO.
		0	No receive packet has been received.
		1	Indicates that a data packet has been received in the RX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.

USBCSRL0 in OTG B/Device mode is shown in Figure 18-35 and described in Table 18-38.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 18-35. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in OTG B/Device Mode

7	6	5	4	3	2	1	0
SETENDC	RXRDYC	STALL	SETEND	DATAEND	STALLED	TXRDY	RXRDY
W1C-0	W1C-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 18-38. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in OTG B/Device Mode Field Descriptions

Bit	Field	Value	Description
7	SETENDC		Setup End Clear
		0	No effect
		1	Writing a 1 to this bit clears the SETEND bit.
6	RXRDYC		RXRDY Clear
		0	No effect
		1	Writing a 1 to this bit clears the RXRDY bit.
5	STALL		Send Stall.
		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		Setup end.
		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		Data end.
		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.
2	STALLED		Endpoint Stalled. Software must clear this bit.
		0	A STALL handshake has not been transmitted.
		1	A STALL handshake has been transmitted.
1	TXRDY		Transmit Packet Ready. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent.
		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.
0	RXRDY		Receive Packet Ready.
		0	No receive 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.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.30 USB Control and Status Endpoint 0 High Register (USBCSRH0), offset 0x103

The USB control and status endpoint 0 high 8-bit register (USBCSRH0) provides control and status bits for endpoint 0.

Mode(s): OTG A or Host OTG B or Device

USBCSRH0 in OTG A/Host mode is shown in Figure 18-36 and described in Table 18-39.

Figure 18-36. USB Control and Status Endpoint 0 High Register (USBCSRH0) in OTG A/Host Mode

7	3	2	1	0
Reserved		DTWE	DT	FLUSH
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-39. USB Control and Status Endpoint 0 High Register (USBCSRH0) in OTG A/Host Mode Field Descriptions

Bit	Field	Value	Description
7-3	Reserved	0	Reserved
2	DTWE		Data Toggle Write Enable. This bit is automatically cleared once the new value is written.
		0	The DT bit cannot be written.
		1	Enables the current state of the endpoint 0 data toggle to be written (see DT bit).
1	DT		Data Toggle. When read, this bit indicates the current state of the endpoint 0 data toggle. If DTWE is set, this bit may be written with the required setting of the data toggle. If DTWE is Low, this bit cannot be written. Care should be taken when writing to this bit as it should only be changed to RESET USB endpoint 0.
0	FLUSH		Flush FIFO. This bit is automatically cleared after the flush is performed.
		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. **Note:** This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

USBCSRH0 in OTG B/Device mode is shown in Figure 18-37 and described in Table 18-40.

Figure 18-37. USB Control and Status Endpoint 0 High Register (USBCSRH0) in OTG B/Device Mode

7	1	0
Reserved		FLUSH
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-40. USB Control and Status Endpoint 0 High Register (USBCSRH0) in OTG B/Device Mode Field Descriptions

Bit	Field	Value	Description
7-1	Reserved	0	Reserved
0	FLUSH		Flush FIFO. This bit is automatically cleared after the flush is performed.
		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. **Note:** This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.31 USB Receive Byte Count Endpoint 0 Register (USBCOUNT0), offset 0x108

The USB receive byte count endpoint 0 8-bit read-only register (USBCOUNT0) 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.

Mode(s): OTG A or Host OTG B or Device

USBCOUNT0 is shown in Figure 18-38 and described in Table 18-30.

Figure 18-38. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0)

7	6	0
Reserved	COUNT	
R-0	R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-41. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	COUNT	0	FIFO Count. COUNT is a read-only value that indicates the number of received data bytes in the endpoint 0 FIFO.

18.5.32 USB Type Endpoint 0 Register (USBTYPE0), offset 0x10A

The USB type endpoint 0 8-bit register (USBTYPE0) must be written with the operating speed of the targeted Device being communicated with using endpoint 0.

Mode(s): OTG A or Host

USBTYPE0 is shown in Figure 18-39 and described in Table 18-42.

Figure 18-39. USB Type Endpoint 0 Register (USBTYPE0)

7	6	5	0
SPEED		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-42. USB Type Endpoint 0 Register (USBTYPE0) Field Descriptions

Bit	Field	Value	Description
7-6	SPEED	0	Operating Speed specifies the operating speed of the target Device. If selected, the target is assumed to have the same connection speed as the USB controller.
		0-1h	Reserved
		2h	Full
		3h	Low
5-0	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.33 USB NAK Limit Register (USBNAKLMT), offset 0x10B

The USB NAK limit 8-bit read-only register (USBNAKLMT) sets the number of frames after which endpoint 0 should time out on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their USBTXINTERVAL[*n*] and USBRXINTERVAL[*n*] registers.)

The number of frames selected is 2(m-1) (where m is the value set in the register, with valid values of 2–16). If the Host receives NAK responses from the target for more frames than the number represented by the limit set in this register, the endpoint is halted.

Note: A value of 0 or 1 disables the NAK timeout function.

Mode(s): OTG A or Host

USBNAKLMT is shown in Figure 18-40 and described in Table 18-43.

Figure 18-40. USB NAK Limit Register (USBNAKLMT)

7	5	4	0
Reserved		NAKLMT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-43. USB NAK Limit Register (USBNAKLMT) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved	0	Reserved
4-0	NAKLMT	0	EP0 NAK Limit specifies the number of frames after receiving a stream of NAK responses.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.34 USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[1]-USBTXCSRL[15])

The USB transmit control and status endpoint *n* low 8-bit registers (USBTXCSRL[*n*]) provide control and status bits for transfers through the currently selected transmit endpoint.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host OTG B or Device

The USBTXCSRL[*n*] registers in OTG A/Host Mode are shown in Figure 18-41 and described in Table 18-44.

Figure 18-41. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in OTG A/Host Mode

7	6	5	4	3	2	1	0
NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-44. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in OTG A/Host Mode Field Descriptions

Bit	Field	Value	Description
7	NAKTO		NAK Timeout. Software must clear this bit to allow the endpoint to continue.
		0	No timeout
		1	Bulk endpoints only: Indicates that the transmit endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBTXINTERVAL[*n*] register.
6	CLRDT		Clear DataToggle
		0	No effect
		1	Writing a 1 to this bit clears the DT bit in the USBTXCSRH[*n*] register.
5	STALLED		Endpoint Stalled. Software must clear this bit.
		0	A STALL handshake has not been received
		1	Indicates that a STALL handshake has been received. When this bit is set, any µDMA request that is in progress is stopped, the FIFO is completely flushed, and the TXRDY bit is cleared.
4	SETUP		Setup Packet.
		0	No SETUP token is sent.
		1	Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.
			Note: Setting this bit also clears the DT bit in the USBTXCSRH[*n*] register.
3	FLUSH		Flush FIFO. This bit can 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.
		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.
			Note: This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted.
2	ERROR		Error. Software must clear this bit.
		0	No error
		1	Three attempts have been made to send a packet and no handshake packet has been received. The TXRDY bit is cleared, the EPn bit in the USBTXIS register is set, and the FIFO is completely flushed in this situation.
			Note: This bit is valid only when the endpoint is operating in Bulk or Interrupt mode.
1	FIFONE		FIFO Not Empty
		0	The FIFO is empty
		1	At least one packet is in the transmit FIFO.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-44. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in OTG A/Host Mode Field Descriptions (continued)

Bit	Field	Value	Description
0	TXRDY		Transmit Packet Ready. This bit is cleared automatically when a data packet has been transmitted. The EP*n* 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.
		0	No transmit packet is ready.
		1	Software sets this bit after loading a data packet into the TX FIFO.

The USBTXCSRL[*n*] registers in OTG B/Device Mode are shown in Table 18-44 and described in Figure 18-42.

Figure 18-42. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in OTG B/Device Mode

7	6	5	4	3	2	1	0
Reserved	CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-45. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in OTG B/Device Mode Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6	CLRDT		Clear Data Toggle
		0	No effect
		1	Writing a 1 to this bit clears the DT bit in the USBTXCSRH[*n*] register.
5	STALLED		Endpoint Stalled. Software must clear this bit.
		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.
4	STALL		Send Stall. Software clears this bit to terminate the STALL condition. **Note:** This bit has no effect in isochronous transfers.
		0	No effect
		1	Issues a STALL handshake to an IN token.
3	FLUSH		Flush FIFO. 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. **Note:** This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted.
		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 is cleared automatically.
2	UNDRN		Underrun. Software must clear this bit.
		0	No underrun
		1	An IN token has been received when TXRDY is not set.
1	FIFONE		FIFO Not Empty
		0	The FIFO is empty.
		1	At least one packet is in the transmit FIFO.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-45. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[*n*]) in OTG B/Device Mode Field Descriptions (continued)

Bit	Field	Value	Description
0	TXRDY		Transmit Packet Ready. 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.
		0	No transmit packet is ready.
		1	Software sets this bit after loading a data packet into the TX FIFO. This bit is cleared by writing a 1 to the RXRDYC bit.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.35 USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[1]-USBTXCSRH[15])

The USB transmit control and status endpoint *n* high 8-bit registers (USBTXCSRH[*n*]) provide additional control for transfers through the currently selected transmit endpoint.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host OTG B or Device

The USBTXCSRH[*n*] registers in OTG A/Host Mode are shown in Figure 18-43 and described in Table 18-46.

Figure 18-43. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in OTG A/Host Mode

7	6	5	4	3	2	1	0
AUTOSET	Reserved	MODE	DMAEN	FDT	DMAMOD	DTWE	DT
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-46. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in OTG A/Host Mode Field Descriptions

Bit	Field	Value	Description
7	AUTOSET		Auto Set
		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 USBTXMAXP[*n*]) 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	Reserved	0	Reserved
5	MODE		Mode **Note:** This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions.
		0	Enables the endpoint direction as RX.
		1	Enables the endpoint direction as TX.
4	DMAEN		DMA Request Enable **Note:** Three TX and three /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.
		0	Disables the µDMA request for the transmit endpoint.
		1	Enables the µDMA request for the transmit endpoint.
3	FDT		Force Data Toggle
		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. **Note:** This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted.
2	DMAMOD		DMA Request Mode **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
		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 is valid only when the endpoint is operating in Bulk or Interrupt mode.
1	DTWE		Data Toggle Write Enable. This bit is automatically cleared once the new value is written.
		0	The DT bit cannot be written.
		1	Enables the current state of the transmit endpoint data to be written (see DT bit).

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-46. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in OTG A/Host Mode Field Descriptions (continued)

Bit	Field	Value	Description
0	DT		Data Toggle. When read, this bit indicates the current state of the transmit endpoint data toggle. If DTWE is High, this bit can be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the transmit endpoint.

The USBTXCSRH[*n*] registers in OTG B/Device Mode are shown in Figure 18-44 and described in Table 18-47.

Figure 18-44. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in OTG B/Device Mode

7	6	5	4	3	2	1	0
AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	Reserved	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 18-47. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[*n*]) in OTG B/Device Mode Field Descriptions

Bit	Field	Value	Description
7	AUTOSET		Auto Set
		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 USBTXMAXP[*n*]) 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		Isochronous Transfers
		0	Enables the transmit endpoint for bulk or interrupt transfers.
		1	Enables the transmit endpoint for isochronous transfers.
5	MODE		Mode **Note:** This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions.
		0	Enables the endpoint direction as RX.
		1	Enables the endpoint direction as TX.
4	DMAEN		DMA Request Enable **Note:** Three TX and three /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.
		0	Disables the µDMA request for the transmit endpoint.
		1	Enables the µDMA request for the transmit endpoint.
3	FDT		Force Data Toggle
		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		DMA Request Mode **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
		0	An interrupt is generated after every µDMA packet transfer.
		1	An interrupt is generated only after the entire µDMA transfer is complete.
0	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.36 USB Maximum Receive Data Endpoint n Registers (USBRXMAXP[1]-USBRXMAXP[15])

The USB maximum receive data endpoint *n* 16-bit registers (USBRXMAXP[*n*]) define 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 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.

Note: USBRXMAXP[*n*] must be set to an even number of bytes for proper interrupt generation in µDMA Basic Mode.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host OTG B or Device

The USBRXMAXP[*n*] registers are shown in Figure 18-45 and described in Table 18-48.

Figure 18-45. USB Maximum Receive Data Endpoint *n* Registers (USBRXMAXP[*n*])

15	11	10	0
Reserved		MAXLOAD	
R-0		R/W-000	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-48. USB Maximum Receive Data Endpoint *n* Registers (USBTXMAXP[*n*]) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10-0	MAXLOAD		Maximum Payload specifies the maximum payload in bytes per transaction.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.37 USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[1]-USBRXCSRL[15])

The USB receive control and status endpoint *n* low 8-bit register (USBCSRL[*n*]) provides control and status bits for transfers through the currently selected receive endpoint.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host OTG B or Device

The USBCSRL[*n*] registers in OTG A/Host mode are shown in Figure 18-46 and described in Table 18-49.

Figure 18-46. USB Receive Control and Status Endpoint *n* Low Register (USBCSRL[*n*]) in OTG A/Host Mode

7	6	5	4	3	2	1	0
CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
W1C-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-49. USB Control and Status Endpoint n Low Register(USBCSRL[*n*]) in OTG A/Host Mode Field Descriptions

Bit	Field	Value	Description
7	NAKTO		Clear Data Toggle.
		0	No effect
		1	Writing a 1 to this bit clears the DT bit in the USBRXCSRH[*n*] register.
6	STALLED		Endpoint Stalled. Software must clear this bit.
		0	No handshake has been received.
		1	A STALL handshake has been received. The EP*n* bit in the USBRXIS register is also set.
5	REQPKT		Request Packet. This bit is cleared when the RXRDY bit is set.
		0	No request
		1	Requests an IN transaction.
4	FLUSH		Flush FIFO. If the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. **Note:**This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted.
		0	No effect
		1	Flushes the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared.
3	DATAERR / NAKTO		Data Error / NAK Timeout
		0	Normal operation
		1	Isochronous endpoints only: Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared when RXRDY is cleared. Bulk endpoints only: Indicates that the receive endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBRXINTERVAL[*n*] register. Software must clear this bit to allow the endpoint to continue.
2	ERROR		Error. Software must clear this bit. **Note:** This bit is only valid when the receive endpoint is operating in Bulk or Interrupt mode. In Isochronous mode, it always returns zero.
		0	No error
		1	Three attempts have been made to receive a packet and no data packet has been received. The EP*n* bit in the USBRXIS register is set in this situation.
1	FULL		FIFO Full
		0	The receive FIFO is not full.
		1	No more packets can be loaded into the receive FIFO.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-49. USB Control and Status Endpoint n Low Register(USBCSRL[*n*]) in OTG A/Host Mode Field Descriptions (continued)

Bit	Field	Value	Description
0	RXRDY		Receive Packet Ready. If the AUTOCLR bit in the USBRXCSRH[*n*] register is set, then the this bit is automatically cleared when a packet of USBRXMAXP[*n*] 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.
		0	No data packet has been received.
		1	Indicates that a data packet has been received. The EP*n* bit in the USBTXIS register is also set in this situation.

USBCSRL0 in OTG B/Device mode is shown in Figure 18-47 and described in Table 18-50.

Figure 18-47. USB Control and Status Endpoint *n* Low Register (USBCSRL[*n*]) in OTG B/Device Mode

7	6	5	4	3	2	1	0
CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
W1C-0	W1C-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-50. USB Control and Status Endpoint 0 Low Register(USBCSRL[*n*]) in OTG B/Device Mode Field Descriptions

Bit	Field	Value	Description
7	CLRDT		Clear Data Toggle
		0	No effect
		1	Writing a 1 to this bit clears the DT bit in the USBRXCSRH[*n*] register.
6	STALLED		Endpoint Stalled. Software must clear this bit.
		0	A STALL handshake has been transmitted.
		1	A STALL handshake has been transmitted.
5	STALL		Send Stall. 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.
		0	No effect
		1	Issues a STALL handshake.
4	FLUSH		Flush FIFO. 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.
		0	No effect
		1	Flushes the next packet from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. **Note:** This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted.
3	DATAEND		Data 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.
		0	Normal operation
		1	Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared automatically.
2	OVER		Overrun. 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.
		0	No overrun error
		1	Indicates an OUT packet cannot be loaded into the receive FIFO.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-50. USB Control and Status Endpoint 0 Low Register(USBCSRL[*n*]) in OTG B/Device Mode Field Descriptions (continued)

Bit	Field	Value	Description
1	FULL		FIFO Full
		0	The receive FIFO is not full.
		1	No more packets can be loaded into the receive FIFO.
0	RXRDY		Receive Packet Ready. If the AUTOCLR bit in the USBRXCSRH[*n*] register is set, then the this bit is automatically cleared when a packet of USBRXMAXP[*n*] 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.
		0	No data packet has been received.
		1	A data packet has been received. The EP*n* bit in the USBTXIS register is also set in this situation. This bit is cleared by writing a 1 to the RXRDYC bit.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.38 USB Receive Control and Status Endpoint n High Register (USBRXCSRH[1]-USBRXCSRH[15])

The USB receive control and status endpoint *n* high 8-bit register (USBCSRL[*n*]) provides additional control and status bits for transfers through the currently selected receive endpoint.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host OTG B or Device

The USBCSRH[*n*] registers in OTG A/Host mode are shown in Figure 18-48 and described in Table 18-51.

Figure 18-48. USB Receive Control and Status Endpoint *n* High Register (USBCSRH[*n*]) in OTG A/Host Mode

7	6	5	4	3	2	1	0
AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	Reserved
W1C-0	R/W-0	R/W-0	R-0	R/W-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-51. USB Control and Status Endpoint n High Register (USBCSRH[*n*]) in OTG A/Host Mode Field Descriptions

Bit	Field	Value	Description
7	AUTOCL		Auto Clear
		0	No effect
		1	Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXP[*n*] 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 USBRXMAXP[*n*] register, see Section 18.2.4.
6	AUTORQ		Auto Request **Note:** This bit is automatically cleared when a short packet is received.
		0	No effect
		1	Enables the REQPKT bit to be automatically set when the RXRDY bit is cleared.
5	DMAEN		DMA Request Enable **Note:** Three TX and three 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.
		0	Disables the µDMA request for the receive endpoint.
		1	Enables the µDMA request for the receive endpoint.
4	PIDERR		PID Error. This bit is ignored in bulk or interrupt transactions.
		0	No error
		1	Indicates a PID error in the received packet of an isochronous transaction.
3	DMAMOD		DMAMOD **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
		0	An interrupt is generated after every µDMA packet transfer.
		1	An interrupt is generated only after the entire µDMA transfer is complete.
2	DTWE		Data Toggle Write Enable. This bit is automatically cleared once the new value is written.
		0	The DT bit cannot be written.
		1	Enables the current state of the receive endpoint data to be written (see DT bit).
1	DT		Data Toggle. When read, this bit indicates the current state of the receive data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the receive endpoint.
0	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

The USBCSRH[*n*] registers in OTG B/Device mode are shown in Figure 18-49 and described in Table 18-52.

Figure 18-49. USB Control and Status Endpoint *n* High Register (USBCSRH[*n*]) in OTG B/Device Mode

7	6	5	4	3	2		0
AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD		Reserved	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-52. USB Control and Status Endpoint 0 High Register(USBCSRH[*n*]) in OTG B/Device Mode Field Descriptions

Bit	Field	Value	Description
7	AUTOCL		Auto Clear
		0	No effect
		1	Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXP[*n*] 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 USBRXMAXP[*n*] register, see Section 18.2.4.
6	ISO		Isochronous Transfers
		0	Enables the receive endpoint for isochronous transfers.
		1	Enables the receive endpoint for bulk/interrupt transfers.
5	DMAEN		DMA Request Enable **Note:** Three TX and three 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.
		0	Disables the µDMA request for the receive endpoint.
		1	Enables the µDMA request for the receive endpoint.
4	DISNYET/PIDERR		Disable NYET / PID Error
		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		DMA Request Mode **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
		0	An interrupt is generated after every µDMA packet transfer.
		1	An interrupt is generated only after the entire µDMA transfer is complete.
0	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.39 USB Receive Byte Count Endpoint n Registers (USBRXCOUNT[1]-USBRXCOUNT[15])

The USB receive byte count endpoint *n* 16-bit read-only registers hold 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.

Note: The value returned changes as the FIFO is unloaded and is only valid while the RXRDY bit in the USBRXCSRLn register is set.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host OTG B or Device

The USBRXCOUNT[*n*] registers are shown in Figure 18-50 and described in Table 18-53.

Figure 18-50. USB Maximum Receive Data Endpoint *n* Registers (USBRXCOUNT[*n*])

15 13	12 0
Reserved	COUNT
R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-53. USB Maximum Receive Data Endpoint *n* Registers (USBRXCOUNT[*n*]) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12-0	COUNT		Receive Packet Count indicates the number of bytes in the receive packet.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.40 USB Host Transmit Configure Type Endpoint n Register (USBTXTYPE[1]-USBTXTYPE[15])

The USB host transmit configure type endpoint *n* 8-bit registers (USBTXTYPE[*n*]) must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected transmit endpoint, and its operating speed.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBTXTYPE[*n*] registers are shown in Figure 18-51 and described in Table 18-54.

Figure 18-51. USB Host Transmit Configure Type Endpoint n Register (USBTXTYPE[*n*])

7	6	5	4	3	0
SPEED		PROTO		TEP	
R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-54. USB Host Transmit Configure Type Endpoint n Register(USBTXTYPE[*n*]) Field Descriptions

Bit	Field	Value	Description
7-6	SPEED		Operating Speed. This bit field specifies the operating speed of the target Device:
		0h	Default. The target is assumed to be using the same connection speed as the USB controller.
		1h	Reserved
		2h	Full
		3h	Low
5-4	PROTO		Protocol. Software must configure this bit field to select the required protocol for the transmit endpoint:
		0h	Control
		1h	Isochronous
		2h	Bulk
		3h	Interrupt
3-0	TEP	0	Target Endpoint Number. Software must configure this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during Device enumeration.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.41 USB Host Transmit Interval Endpoint n Register (USBTXINTERVAL[1]USBTXINTERVAL[15])

The USB host transmit interval endpoint *n* 8-bit registers (USBTXINTERVAL[*n*]), for interrupt and isochronous transfers, define the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The USBTXINTERVAL[*n*] registers values define a number of frames, as follows:

Table 18-55. USBTXINTERVAL[*n*] Frame Numbers

Transfer Type	Speed	Valid Values (m)	Interpretation
Interrupt	Low-speed or Full-speed	0x01-0xFF	The polling interval is *m* frames.
Isochronous	Full-speed	0x01-0x10	The polling interval is $2^{(m-1)}$ frames.
Bulk	Full-speed	0x02-0x10	The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBTXINTERVAL[*n*] registers are shown in Figure 18-51 and described in Table 18-54.

Figure 18-52. USB Host Transmit Interval Endpoint n Register (USBTXINTERVAL[*n*])

7	0
TXPOLL / NAKLMT	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-56. USB Host Transmit Interval Endpoint n Register(USBTXINTERVAL[*n*]) Field Descriptions

Bit	Field	Value	Description
7-0	TXPOLL / NAKLMT	0	TX Polling / NAK Limit The polling interval for interrupt/isochronous transfers; the NAK limit for bulk transfers. See Table 18-55 for valid entries; other values are reserved.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.42 USB Host Configure Receive Type Endpoint n Register (USBRXTYPE[1]-USBRXTYPE[15])

The USB host configure receive type endpoint *n* 8-bit registers (USBRXTYPE[*n*]) must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected receive endpoint, and its operating speed.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBRXTYPE[*n*] registers are shown in Figure 18-53 and described in Table 18-57.

Figure 18-53. USB Host Configure Receive Type Endpoint n Register (USBRXTYPE[*n*])

7	6	5	4	3	0
SPEED		PROTO		TEP	
R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-57. USB Host Configure Receive Type Endpoint n Register(USBRXTYPE[*n*]) Field Descriptions

Bit	Field	Value	Description
7-6	SPEED		Operating Speed. This bit field specifies the operating speed of the target Device:
		0h	Default. The target is assumed to be using the same connection speed as the USB controller.
		1h	Reserved
		2h	Full
		3h	Low
5-4	PROTO		Protocol. Software must configure this bit field to select the required protocol for the receive endpoint:
		0h	Control
		1h	Isochronous
		2h	Bulk
		3h	Interrupt
3-0	TEP	0	Target Endpoint Number. Software must configure this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during Device enumeration.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.43 USB Host Receive Polling Interval Endpoint n Register (USBRXINTERVAL[1]USBRXINTERVAL[15])

The USB host receive polling interval endpoint *n* 8-bit registers (USBRXINTERVAL[*n*]), for interrupt and isochronous transfers, define the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The USBRXINTERVAL[*n*] registers values define a number of frames, as follows:

Table 18-58. USBRXINTERVAL[*n*] Frame Numbers

Transfer Type	Speed	Valid Values (m)	Interpretation
Interrupt	Low-speed or Full-speed	0x01-0xFF	The polling interval is *m* frames.
Isochronous	Full-speed	0x01-0x10	The polling interval is $2^{(m-1)}$ frames.
Bulk	Full-speed	0x02-0x10	The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBRXINTERVAL[*n*] registers are shown in Figure 18-51 and described in Table 18-54.

Figure 18-54. USB Host Receive Polling Interval Endpoint n Register (USBRXINTERVAL[*n*])

7	0
TXPOLL / NAKLMT	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 18-59. USB Host Receive Polling Interval Endpoint n Register(USBRXINTERVAL[*n*]) Field Descriptions

Bit	Field	Value	Description
7-0	TXPOLL / NAKLMT	0	TX Polling / NAK Limit The polling interval for interrupt/isochronous transfers; the NAK limit for bulk transfers. See Table 18-58 for valid entries; other values are reserved.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.44 USB Request Packet Count in Block Transfer Endpoint n Registers (USBRQPKTCOUNT[1]USBRQPKTCOUNT[15])

The USB receive packet count in block transfer endpoint *n* 16-bit read/writer registers are used in Host mode to specify the number of packets that are to be transferred in a block transfer of one or more bulk packets to receive endpoint *n*. The USB controller uses the value recorded in this register to determine the number of requests to issue where the AUTORQ bit in the USBRXCSRH[*n*] register has been set. For more information about IN transactions as a host, see Section 18.2.2.2.

Note: Multiple packets combined into a single bulk packet within the FIFO count as one packet.

For the specific offset for each register see Table 18-4.

Mode(s): OTG A or Host

The USBRQPKTCOUNT[*n*] registers are shown in Figure 18-55 and described in Table 18-60.

Figure 18-55. USB Request Packet Count in Block Transfer Endpoint *n* Registers (USBRQPKTCOUNT[*n*])

15	13	12	0
Reserved		COUNT	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-60. USB Request Packet Count in Block Transfer Endpoint *n* Registers (USBRQPKTCOUNT[*n*]) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12-0	COUNT		Block Transfer Packet Count sets the number of packets of the size defined by the MAXLOAD bit field that are to be transferred in a block transfer. **Note:** This is only used in Host mode when AUTORQ is set. The bit has no effect in Device mode or when AUTORQ is not set.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.45 USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS), offset 0x340

The USB receive double packet buffer disable 16-bit register (USBTXIE) indicates which of the receive endpoints have disabled the double-packet buffer functionality (see *Double-Packet Buffering* in Section 18.2.1.1.1).

Mode(s): OTG A or Host OTG B or Device

USBRXDPKTBUFDIS is shown in Figure 18-56 and described in Table 18-61.

Figure 18-56. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	Rsvd
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-61. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS) Field Descriptions

Bit	Field	Value	Description
15	EP15		EP15 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
14	EP14		EP14 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
13	EP13		EP13 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
12	EP12		EP12 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
11	EP11		EP11 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
10	EP10		EP10 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
9	EP9		EP9 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
8	EP8		EP8 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
7	EP7		EP7 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
6	EP6		EP6 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-61. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS) Field Descriptions (continued)

Bit	Field	Value	Description
5	EP5		EP5 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
4	EP4		EP4 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
3	3P3		EP3 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
2	EP2		EP2 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
1	EP1		EP1 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
0	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.46 USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS), offset 0x342

The USB transmit double packet buffer disable 16-bit register (USBTXIE) indicates which of the transmit endpoints have disabled the double-packet buffer functionality (see *Double-Packet Buffering* in Section 18.2.1.1.1).

Mode(s): OTG A or Host OTG B or Device

USBTXDPKTBUFDIS is shown in Figure 18-57 and described in Table 18-62.

Figure 18-57. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	Rsvd
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-62. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS) Field Descriptions

Bit	Field	Value	Description
15	EP15		EP15 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
14	EP14		EP14 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
13	EP13		EP13 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
12	EP12		EP12 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
11	EP11		EP11 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
10	EP10		EP10 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
9	EP9		EP9 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
8	EP8		EP8 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
7	EP7		EP7 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
6	EP6		EP6 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-62. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS)

Field Descriptions (continued)

Bit	Field	Value	Description
5	EP5		EP5 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
4	EP4		EP4 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
3	3P3		EP3 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
2	EP2		EP2 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
1	EP1		EP1 RX Double-Packet Buffer Disable
		0	Disables double-packet buffering.
		1	Enables double-packet buffering.
0	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.47 USB External Power Control Register (USBEPC), offset 0x400

The USB external power control 32-bit register (USBEPC) specifies the function of the two-pin external power interface (USB0EPEN and USB0PFLT). The assertion of the power fault input may generate an automatic action, as controlled by the hardware configuration registers. The automatic action is necessary because the fault condition may require a response faster than one provided by firmware.

Mode(s): OTG A or Host OTG B or Device

USBEPC is shown in Figure 18-58 and described in Table 18-63.

Figure 18-58. USB External Power Control Register (USBEPC)

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

15	14	13	12	11	10	9	8
Reserved						PFLTACT	
R-0						R/W-0	

7	6	5	4	3	2	1	0
Reserved	PFLTAEN	PFLTSEN	PFLTEN	Reserved	EPENDE	EPEN	
R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 18-63. USB External Power Control Register (USBEPC) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-8	PFLTACT		Power Fault Action. This bit field specifies how the USB0EPEN signal is changed when detecting a USB power fault.
		0h	Unchanged. USB0EPEN is controlled by the combination of the EPEN and EPENDE bits.
		1h	Tristate. USB0EPEN is undriven (tristate).
		2h	Low. USB0EPEN is driven Low.
		3h	High. USB0EPEN is driven High.
7	Reserved	0	Reserved
6	PFLTAEN		Power Fault Action Enable. This bit specifies whether a USB power fault triggers any automatic corrective action regarding the driven state of the USB0EPEN signal.
		0	Disabled. USB0EPEN is controlled by the combination of the EPEN and EPENDE bits.
		1	Enabled. The USB0EPEN output is automatically changed to the state specified by the PFLTACT field.
5	PFLTSEN		Power Fault Sense. This bit specifies the logical sense of the USB0PFLT input signal that indicates an error condition. The complementary state is the inactive state.
		0	Low Fault. If USB0PFLT is driven Low, the power fault is signaled internally (if enabled by the PFLTEN bit).
		1	High Fault. If USB0PFLT is driven High, the power fault is signaled internally (if enabled by the PFLTEN bit).
4	PFLTEN		Power Fault Input Enable. This bit specifies whether the USB0PFLT input signal is used in internal logic.
		0	Not Used. The USB0PFLT signal is ignored.
		1	Used. The USB0PFLT signal is used internally
3	Reserved	0	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-63. USB External Power Control Register (USBEPC) Field Descriptions (continued)

Bit	Field	Value	Description
2	EPENDE		EPEN Drive Enable. This bit specifies whether the USB0EPEN signal is driven or undriven (tristate). When driven, the signal value is specified by the EPEN field. When not driven, the EPEN field is ignored and the USB0EPEN signal is placed in a high-impedance state. The USB0EPEN signal is undriven at reset because the sense of the external power supply enable is unknown. By adding the high-impedance state, system designers can bias the power supply enable to the disabled state using a large resistor (100 kΩ) and later configure and drive the output signal to enable the power supply.
		0	Not Driven. The USB0EPEN signal is high impedance.
		1	Driven. The USB0EPEN signal is driven to the logical value specified by the value of the EPEN field.
1-0	EPEN		External Power Supply Enable Configuration. This bit field specifies and controls the logical value driven on the USB0EPEN signal.
		0h	Power Enable Active Low. The USB0EPEN signal is driven Low if the EPENDE bit is set.
		1h	Power Enable Active High. The USB0EPEN signal is driven High if the EPENDE bit is set.
		2h	Power Enable High if VBUS Low. The USB0EPEN signal is driven High when the A device is not recognized.
		3h	Power Enable High if VBUS High. The USB0EPEN signal is driven High when the A device is recognized.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.48 USB External Power Control Raw Interrupt Status Register (USBEPCRIS), offset 0x404

The USB external power control raw interrupt status 32-bit register (USBEPCRIS) specifies the unmasked interrupt status of the two-pin external power interface.

Mode(s): OTG A or Host OTG B or Device

USBEPCRIS is shown in Figure 18-59 and described in Table 18-64.

Figure 18-59. USB External Power Control Raw Interrupt Status Register (USBEPCRIS)

31	1	0
Reserved		PF
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-64. USB External Power Control Raw Interrupt Status Register (USBEPCRIS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	PF		USB Power Fault Interrupt Status. This bit is cleared by writing a 1 to the PF bit in the USBEPCISC register.
		0	A Power Fault status has been detected.
		1	An interrupt has not occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.49 USB External Power Control Interrupt Mask Register (USBEPCIM), offset 0x408

The USB external power control interrupt mask 32-bit register (USBEPCIM) specifies the interrupt mask of the two-pin external power interface.

Mode(s): OTG A or Host OTG B or Device

USBEPCIM is shown in Figure 18-59 and described in Table 18-64.

Figure 18-60. USB External Power Control Interrupt Mask Register (USBEPCIM)

31	1	0
Reserved		PF
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-65. USB External Power Control Interrupt Mask Register (USBEPCIM) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	PF		USB Power Fault Interrupt Mask.
		0	The raw interrupt signal from a detected power fault is sent to the interrupt controller.
		1	A detected power fault does not affect the interrupt status.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.50 USB External Power Control Interrupt Status and Clear Register (USBEPCISC), offset 0x40C

The USB external power control interrupt status and clear 32-bit register (USBEPCISC) specifies the unmasked interrupt status of the two-pin external power interface.

Mode(s): OTG A or Host OTG B or Device

USBEPCISC is shown in Figure 18-61 and described in Table 18-66.

Figure 18-61. USB External Power Control Interrupt Status and Clear Register (USBEPCISC)

31	1	0
Reserved		PF
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-66. USB External Power Control Interrupt Status and Clear Register (USBEPCISC) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	PF		USB Power Fault Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the PF bit in the USBEPCISC register.
		0	The PF bits in the USBEPCRIS and USBEPCIM registers are set, providing an interrupt to the interrupt controller.
		1	No interrupt has occurred or the interrupt is masked.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.51 USB Device RESUME Raw Interrupt Status Register (USBDRRIS), offset 0x410

The USB device RESUME raw interrupt status register (USBDRRIS) 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.

Mode(s): OTG A or Host OTG B or Device

USBDRRIS is shown in Figure 18-62 and described in Table 18-67.

Figure 18-62. USB Device RESUME Raw Interrupt Status Register (USBDRRIS)

31	1	0
Reserved		RESUME
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-67. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	PF		RESUME Interrupt Status This bit is cleared by writing a 1 to the RESUME bit in the USBDRISC register.
		0	A RESUME status has been detected.
		1	An interrupt has not occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.52 USB Device RESUME Raw Interrupt Mask Register (USBDRIM), offset 0x414

The USB device RESUME raw interrupt status register (USBDRIM) 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.

Mode(s): OTG A or Host OTG B or Device

USBDRIM is shown in Figure 18-63 and described in Table 18-68.

Figure 18-63. USB Device RESUME Raw Interrupt Status Register (USBDRRIS)

31	1	0
Reserved		RESUME
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-68. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	PF		RESUME Interrupt Mask
		0	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).
		1	A detected RESUME does not affect the interrupt status.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.53 USB Device RESUME Interrupt Status and Clear Register (USBDRISC), offset 0x418

The USB device RESUME interrupt status and clear register (USBDRRIS) is the raw interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

Mode(s): OTG A or Host OTG B or Device

USBDRISC is shown in Figure 18-64 and described in Table 18-69.

Figure 18-64. USB Device RESUME Interrupt Status and Clear Register (USBDRISC)

31	0
Reserved	RESUME
R-0	R/W1C

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-69. USB Device RESUME Interrupt Status and Clear Register (USBDRISC) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	RESUME		RESUME Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the RESUME bit in the USBDRCRIS register.
		0	The RESUME bits in the USBDRRIS and USBDRCIM registers are set, providing an interrupt to the interrupt controller.
		1	No interrupt has occurred or the interrupt is masked.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.54 USB General-Purpose Control and Status Register (USBGPCS), offset 0x41C

The USB general-purpose control and status register (USBGPCS) provides the state of the internal ID signal.

When used in OTG mode, USB0VBUS and USB0ID do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector's VBUS and ID signals. If the USB controller is used as either a dedicated Host or Device, the DEVMODOTG and DEVMOD bits in the USB General-Purpose Control and Status (USBGPCS) register can be used to connect the USB0VBUS and USB0ID inputs to fixed levels internally, freeing the pins for GPIO use. For proper self-powered Device operation, the VBUS value must still be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pull-up resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

Mode(s): OTG A or Host OTG B or Device

USBGPCS is shown in Figure 18-65 and described in Table 18-70.

Figure 18-65. USB General-Purpose Control and Status Register (USBGPCS)

31	2	1	0
Reserved		DEVMODOTG	DEVMOD
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-70. USB General-Purpose Control and Status Register (USBGPCS) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Reset is 0x0000.000.
1	DEVMODOTG		Enable Device Mode. This bit enables the DEVMOD bit to control the state of the internal ID signal in OTG mode.
		0	The mode is specified by the state of the internal ID signal.
		1	This bit enables the DEVMOD bit to control the internal ID signal.
0	DEVMOD		Device Mode This bit specifies the state of the internal ID signal in Host mode and in OTG mode when the DEVMODOTG bit is set. In Device mode this bit is ignored (assumed set).
		0	Host mode
		1	Device mode

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.55 USB VBUS Droop Control Register (USBVDC), offset 0x430

The USB VBUS droop control 32-bit register (USBVDC) enables a controlled masking of VBUS to compensate for any in-rush current by a Device that is connected to the Host controller. The in-rush current can cause VBUS to droop, causing the USB controller's behavior to be unexpected. The USB Host controller allows VBUS to fall lower than the VBUS Valid level (4.75 V) but not below AValid (2.0 V) for 65 microseconds without signaling a VBUSERR interrupt in the controller. Without this, any glitch on VBUS would force the USB Host controller to remove power from VBUS and then re-enumerate the Device.

Mode(s): OTG A or Host

USBVDC is shown in Figure 18-66 and described in Table 18-71.

Figure 18-66. USB VBUS Droop Control Register (USBVDC)

31	0
Reserved	VBDEN
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-71. USB VBUS Droop Control Register (USBVDC) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	VBDEN		VBUS Droop Enable
		0	No effect
		1	Any changes from VBUSVALID are masked when VBUS goes below 4.75 V but not lower than 2.0 V for 65 microseconds. During this time, the VBUS state indicates VBUSVALID.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.56 USB VBUS Droop Control Raw Interrupt Status Register (USBVDCRIS), offset 0x434

The USB VBUS droop control raw interrupt status 32-bit register (USBVDCRIS) specifies the unmasked interrupt status of the VBUS droop limit of 65 microseconds.

Mode(s): OTG A or Host

USBVDCRIS is shown in Figure 18-67 and described in Table 18-72.

Figure 18-67. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCRIS)

31	1	0
Reserved		VD
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-72. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCRIS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	VD		VBUS Droop Raw Interrupt Status. This bit is cleared by writing a 1 to the VD bit in the USBVDCISC register.
		0	A VBUS droop lasting for 65 microseconds has been detected.
		1	An interrupt has not occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.57 USB VBUS Droop Control Interrupt Mask Register (USBVDCIM), offset 0x438

The USB VBUS droop control interrupt mask 32-bit register (USBVDCIM) specifies the interrupt mask of the VBUS droop.

Mode(s): OTG A or Host

USBVDCIM is shown in Figure 18-68 and described in Table 18-73.

Figure 18-68. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCIM)

31	1	0
Reserved		VD
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-73. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCIM) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	VD		VBUS Droop Interrupt Mask.
		0	The raw interrupt signal from a detected VBUS droop is sent to the interrupt controller.
		1	A detected VBUS droop does not affect the interrupt status.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.58 USB VBUS Droop Control Interrupt Status and Clear Register (USBVDCISC), offset 0x43C

The USB VBUS droop control interrupt status and clear 32-bit register (USBVDCRIS) specifies the masked interrupt status of the VBUS droop and provides a method to clear the interrupt state.

Mode(s): OTG A or Host

USBVDCISC is shown in Figure 18-69 and described in Table 18-74.

Figure 18-69. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCISC)

31	1	0
Reserved		VD
R-0		R/W1C

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-74. USB VBUS Droop Control Raw Interrupt Status Register (USBVDCISC) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	VD		VBUS Droop Interrupt Status and Clear This bit is cleared by writing a 1. Clearing this bit also clears the VD bit in the USBVDCRIS register.
		0	The VD bits in the USBVDCRIS and USBVDCIM registers are set, providing an interrupt to the interrupt controller.
		1	No interrupt has occurred or the interrupt is masked.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.59 USB ID Valid Detect Raw Interrupt Status Register (USBIDVRIS), offset 0x444

The USB ID valid detect raw interrupt status 32-bit register (USBIDVRIS) specifies whether the unmasked interrupt status of the ID value is valid.

Mode(s): OTG-specific functions

USBIDVRIS is shown in Figure 18-70 and described in Table 18-75.

Figure 18-70. USB ID Valid Detect Raw Interrupt Status Register (USBIDVRIS)

31	1	0
Reserved		ID
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-75. USB ID Valid Detect Raw Interrupt Status Register (USBIDVRIS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	VD		ID Valid Detect Raw Interrupt Status This bit is cleared by writing a 1 to the ID bit in the USBIDVISC register.
		0	A valid ID has been detected.
		1	An interrupt has not occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.60 USB ID Valid Detect Interrupt Mask Register (USBIDVIM), offset 0x448

The USB ID valid detect interrupt mask 32-bit register (USBIDVIM) specifies the interrupt mask of the ID valid detection.

Mode(s): OTG-specific functions

USBIDVIM is shown in Figure 18-71 and described in Table 18-76.

Figure 18-71. USB ID Valid Detect Interrupt Mask Register (USBIDVIM)

31	1	0
Reserved		ID
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-76. USB ID Valid Detect Interrupt Mask Register (USBIDVIM) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	VD		ID Valid Detect Interrupt Mask
		0	The raw interrupt signal from a detected ID valid is sent to the interrupt controller.
		1	A detected ID valid does not affect the interrupt status.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.61 USB ID Valid Detect Interrupt Status and Clear Register (USBIDVISC), offset 0x44C

The USB ID valid detect interrupt status and clear 32-bit register (USBIDVISC) specifies whether the unmasked interrupt status of the ID value is valid.

Mode(s): OTG-specific functions

USBIDVISC is shown in Figure 18-72 and described in Table 18-77.

Figure 18-72. USB ID Valid Detect Interrupt Status and Clear Register (USBIDVISC)

31	8	0
Reserved		ID
R-0		R/W1 C-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-77. USB ID Valid Detect Interrupt Status and Clear Register (USBIDVISC) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	VD		ID Valid Detect Raw Interrupt Status This bit is cleared by writing a 1. Clearing this bit also clears the ID bit in the USBIDVRIS register.
		0	The ID bits in the USBIDVRIS and USBIDVIM registers are set, providing an interrupt to the interrupt controller.
		1	No interrupt has occurred or the interrupt is masked.

Copyright © 2012–2019, Texas Instruments Incorporated

18.5.62 USB DMA Select Register (USBDMASEL), offset 0x450

The USB DMA select 32-bit register (USBDMASEL) specifies whether the unmasked interrupt status of the ID value is valid.

Mode(s): OTG-specific functions

USBDMASEL is shown in Figure 18-73 and described in Table 18-77.

Figure 18-73. USB DMA Select Register (USBDMASEL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								DMACTX				DMACRX			
R/0								R/W-0				R/W-0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMABTX				DMABRX				DMAATX				DMAARX			
R/W-0				R/W-0				R/W-0				R/W-0			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18-78. USB DMA Select Register (USBDMASEL) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved. Reset is 0x0000.000.
23-20	DMACTX		DMA C TX Select specifies the TX mapping of the third USB endpoint on µDMA channel 5 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 TX
		2h	Endpoint 2 TX
		3h	Endpoint 3 TX
		4h	Endpoint 4 TX
		5h	Endpoint 5 TX
		6h	Endpoint 6 TX
		7h	Endpoint 7 TX
		8h	Endpoint 8 TX
		9h	Endpoint 9 TX
		Ah	Endpoint 10 TX
		Bh	Endpoint 11 TX
		Ch	Endpoint 12 TX
		Dh	Endpoint 13 TX
		Eh	Endpoint 14 TX
		Fh	Endpoint 15 TX

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-78. USB DMA Select Register (USBDMASEL) Field Descriptions (continued)

Bit	Field	Value	Description
19-16	DMACRX		DMA C RX Select specifies the RX and TX mapping of the third USB endpoint on µDMA channel 4 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 RX
		2h	Endpoint 2 RX
		3h	Endpoint 3 RX
		4h	Endpoint 4 RX
		5h	Endpoint 5 RX
		6h	Endpoint 6 RX
		7h	Endpoint 7 RX
		8h	Endpoint 8 RX
		9h	Endpoint 9 RX
		Ah	Endpoint 10 RX
		Bh	Endpoint 11 RX
		Ch	Endpoint 12 RX
		Dh	Endpoint 13 RX
		Eh	Endpoint 14 RX
		Fh	Endpoint 15 RX
15-12	DMABTX		DMA B TX Select specifies the TX mapping of the second USB endpoint on µDMA channel 3 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 TX
		2h	Endpoint 2 TX
		3h	Endpoint 3 TX
		4h	Endpoint 4 TX
		5h	Endpoint 5 TX
		6h	Endpoint 6 TX
		7h	Endpoint 7 TX
		8h	Endpoint 8 TX
		9h	Endpoint 9 TX
		Ah	Endpoint 10 TX
		Bh	Endpoint 11 TX
		Ch	Endpoint 12 TX
		Dh	Endpoint 13 TX
		Eh	Endpoint 14 TX
		Fh	Endpoint 15 TX

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-78. USB DMA Select Register (USBDMASEL) Field Descriptions (continued)

Bit	Field	Value	Description
11-8	DMABRX		DMA B RX Select Specifies the RX mapping of the second USB endpoint on μDMA channel 2 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 RX
		2h	Endpoint 2 RX
		3h	Endpoint 3 RX
		4h	Endpoint 4 RX
		5h	Endpoint 5 RX
		6h	Endpoint 6 RX
		7h	Endpoint 7 RX
		8h	Endpoint 8 RX
		9h	Endpoint 9 RX
		Ah	Endpoint 10 RX
		Bh	Endpoint 11 RX
		Ch	Endpoint 12 RX
		Dh	Endpoint 13 RX
		Eh	Endpoint 14 RX
		Fh	Endpoint 15 RX
7-4	DMAATX		DMA A TX Select specifies the TX mapping of the first USB endpoint on μDMA channel 1 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 TX
		2h	Endpoint 2 TX
		3h	Endpoint 3 TX
		4h	Endpoint 4 TX
		5h	Endpoint 5 TX
		6h	Endpoint 6 TX
		7h	Endpoint 7 TX
		8h	Endpoint 8 TX
		9h	Endpoint 9 TX
		Ah	Endpoint 10 TX
		Bh	Endpoint 11 TX
		Ch	Endpoint 12 TX
		Dh	Endpoint 13 TX
		Eh	Endpoint 14 TX
		Fh	Endpoint 15 TX

Copyright © 2012–2019, Texas Instruments Incorporated

Table 18-78. USB DMA Select Register (USBDMASEL) Field Descriptions (continued)

Bit	Field	Value	Description
3-0	DMAARX		DMA A RX Select specifies the RX mapping of the first USB endpoint on µDMA channel 0 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 RX
		2h	Endpoint 2 RX
		3h	Endpoint 3 RX
		4h	Endpoint 4 RX
		5h	Endpoint 5 RX
		6h	Endpoint 6 RX
		7h	Endpoint 7RX
		8h	Endpoint 8RX
		9h	Endpoint 9 RX
		Ah	Endpoint 10 RX
		Bh	Endpoint 11 RX
		Ch	Endpoint 12 RX
		Dh	Endpoint 13 RX
		Eh	Endpoint 14 RX
		Fh	Endpoint 15 RX

Copyright © 2012–2019, Texas Instruments Incorporated

M3 Ethernet Media Access Controller (EMAC)

The Ethernet Media Access Controller (EMAC) conforms to IEEE 802.3 specifications and fully supports 10BASE-T and 100BASE-TX standards.

Topic **Page**

19.1 Introduction **1381**
19.2 EMAC Block Diagram **1381**
19.3 Functional Description **1382**
19.4 Initialization and Configuration **1386**
19.5 Register Map **1387**
19.6 Ethernet MAC Register Descriptions **1388**
19.7 MII Management Register Descriptions **1402**

Copyright © 2012–2019, Texas Instruments Incorporated

19.1 Introduction

The EMAC module has the following features:

- Conforms to the IEEE 802.3-2002 specification
  - 10BASE-T/100BASE-TX IEEE-802.3 compliant.
- Multiple operational modes
  - Full- and half-duplex 100 Mbps
  - Full- and half-duplex 10 Mbps
  - Power-saving and power-down modes
- Highly configurable
  - Programmable MAC address
  - Promiscuous mode support
  - CRC error-rejection control
  - User-configurable interrupts
- IEEE 1588 Precision Time Protocol: Provides highly accurate time stamps for individual packets
- Efficient transfers using the Micro Direct Memory Access Controller (μDMA)
  - Separate channels for transmit and receive
  - Receive channel request asserted on packet receipt
  - Transmit channel request asserted on empty transmit FIFO

19.2 EMAC Block Diagram

As shown in Figure 19-1, the MAC layer corresponds to the OSI model layer 2. The MAC layer provides transmit and receive processing for Ethernet frames. It also provides the interface to the physical layer (PHY) via an internal media independent interface (MII). The PHY layer communicates with the Ethernet bus.

Figure 19-1. Ethernet MAC

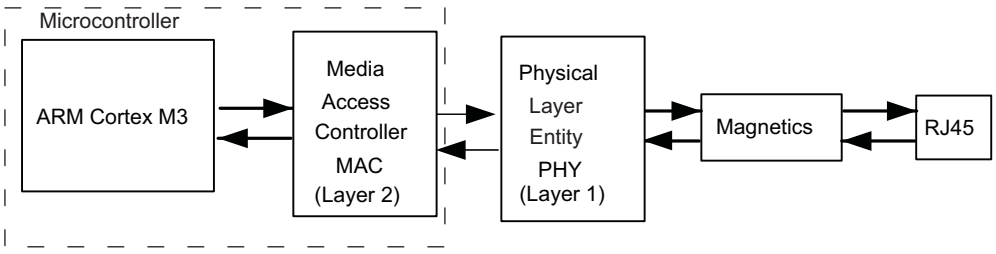
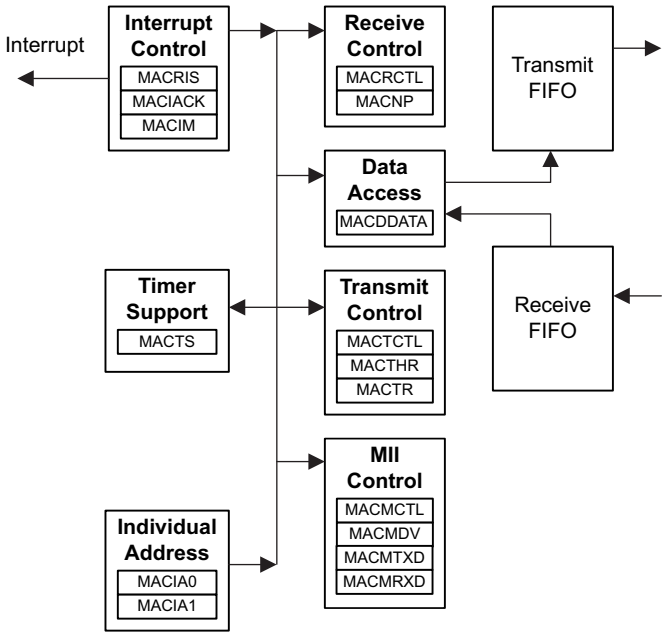


Figure 19-2 shows more detail of the internal structure of the EMAC and how the register set relates to various functions.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 19-2. Ethernet MAC Block Diagram



19.3 Functional Description

The functional description of the EMAC is discussed in the following sections.

19.3.1 MAC Operation

The following sections describe the operation of the MAC layer, including an overview of the Ethernet frame format, the MAC layer FIFOs, Ethernet transmission, reception options, packet timestamps, and Ethernet MAC address format.

19.3.1.1 Ethernet Frame Format

Ethernet data is carried by Ethernet frames. The basic frame format is shown in Figure 19-3.

Figure 19-3. Ethernet Frame

Preamble	SFD	Destination Address	Source Address	Length/ Type	Data	FCS
7 Bytes	1 Byte	6 Bytes	6 Bytes	2 Bytes	46 -1500 Bytes	4 Bytes

The seven fields of the frame are transmitted from left to right. The bits within the frame are transmitted from least to most significant bit.

- **Preamble**
  The Preamble field is used to synchronize with the received frame's timing. The preamble is 7 octets long.
- **Start Frame Delimiter (SFD)**
  The SFD field follows the preamble pattern and indicates the start of the frame. Its value is 1010.1011b.
- **Destination Address (DA)**
  This field specifies destination addresses for which the frame is intended. The LSB (bit 16 of DA oct 1 in the frame, see Table 18-3 on page 907) of the DA determines whether the address is an individual (0), or group/multicast (1) address.

Copyright © 2012–2019, Texas Instruments Incorporated

- **Source Address (SA)**
  The source address field identifies the station from which the frame was initiated..
- **Length/Type Field**
  The meaning of this field depends on its numeric value. This field can be interpreted as length or type code. The maximum length of the data field is 1500 octets. If the value of the Length/Type field is less than or equal to 1500 decimal, it indicates the number of MAC client data octets. If the value of this field is greater than or equal to 1536 decimal, then it encodes the type interpretation. The meaning of the Length/Type field when the value is between 1500 and 1536 decimal is unspecified by the IEEE 802.3 standard. However, the Ethernet MAC assumes type interpretation if the value of the Length/Type field is greater than 1500 decimal. The definition of the Type field is specified in the IEEE 802.3 standard. The first of the two octets in this field is most significant.
- **Data**
  The data field is a sequence of octets that is at least 46 in length, up to 1500 in length. Full data transparency is provided so any values can appear in this field. A minimum frame size of 46 octets is required to meet the IEEE standard. If the frame size is too small, the Ethernet MAC automatically appends extra bits (a pad), thus the pad can have a size of 0 to 46 octets. Data padding can be disabled by clearing the PADEN bit in the Ethernet MAC Transmit Control (MACTCTL) register.
  For the Ethernet MAC, data sent/received can be larger than 1500 bytes without causing a Frame Too Long error. Instead, a FIFO overrun error is reported using the FOV bit in the Ethernet MAC Raw Interrupt Status (MACRIS) register when the frame received is too large to fit into the Ethernet MAC's 2K RAM.
- **Frame Check Sequence (FCS)**
  The frame check sequence carries the cyclic redundancy check (CRC) value. The CRC is computed over the destination address, source address, length/type, and data (including pad) fields using the CRC-32 algorithm. The Ethernet MAC computes the FCS value one nibble at a time. For transmitted frames, this field is automatically inserted by the MAC layer, unless disabled by clearing the CRC bit in the MACTCTL register. For received frames, this field is automatically checked. If the FCS does not pass, the frame is not placed in the RX FIFO, unless the FCS check is disabled by clearing the BADCRC bit in the MACRCTL register.

19.3.1.2 MAC Layer FIFOs

The Ethernet MAC is capable of simultaneous transmission and reception. This feature is enabled by setting the DUPLEX bit in the MACTCTL register.

For Ethernet frame transmission, a 2-KB transmit FIFO is provided that can be used to store a single frame. While the IEEE 802.3 specification limits the size of an Ethernet frame's payload section to 1500 Bytes, the Ethernet MAC places no such limit. The full buffer can be used for a payload of up to 2032 bytes (as the first 16 bytes in the FIFO are reserved for destination address, source address and length/type information).

For Ethernet frame reception, a 2-KB receive FIFO is provided that can be used to store multiple frames, up to a maximum of 31 frames. If a frame is received, and there is insufficient space in the RX FIFO, an overflow error is indicated using the FOV bit in the MACRIS register.

For details regarding the TX and RX FIFO layout, refer to Table 19-1. Please note the following difference between TX and RX FIFO layout. For the TX FIFO, the Data Length field in the first FIFO word refers to the Ethernet frame data payload, as shown in the 5th to nth FIFO positions. For the RX FIFO, the Frame Length field is the total length of the received Ethernet frame, including the Length/Type bytes and the FCS bits.

If FCS generation is disabled by clearing the CRC bit in the MACTCTL register, the last word in the TX FIFO must contain the FCS bytes for the frame that has been written to the FIFO.

Also note that if the length of the data payload section is not a multiple of 4, the FCS field is not be aligned on a word boundary in the FIFO. However, for the RX FIFO, the beginning of the next frame is always on a word boundary.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 19-1. TX & RX FIFO Organization

FIFO Word Read/Write Sequence	Word Bit Fields	TX FIFO (Write)	RX FIFO (Read)
1st	1:0	Data Length Least Significant Byte	Frame Length Least Significant Byte
	15:8	Data Length Most Significant Byte	Frame Length Most Significant Byte
	23:16	DA oct 1	
	31:24	DA oct 2	
2nd	7:0	DA oct 3	
	15:8	DA oct 4	
	23:16	DA oct 5	
	31:24	DA oct 6	
3rd	7:0	SA oct 1	
	15:8	SA oct 2	
	23:16	SA oct 3	
	31:24	SA oct 4	
4th	7:0	SA oct 5	
	15:8	SA oct 6	
	23:16	Len/Type Most Significant Byte	
	31:24	Len/Type Least Significant Byte	
5th to nth	7:0	data oct n	
	15:8	data oct n + 1	
	23:16	data oct n + 2	
	31:24	data oct n + 3	
last	7:0	FCS 1 (1)	
	15:8	FCS 2 (1)	
	23:16	FCS 3 (1)	
	31:24	FCS 4 (1)	

(1) If the CRC bit in the MACTCTL register is clear, the FCS bytes must be written with the correct CRC. If the CRC bit is set, the Ethernet MAC automatically writes the FCS bytes.

19.3.1.3 Ethernet Transmission Options

At the MAC layer, the transmitter can be configured for both full-duplex and half-duplex operation by using the DUPLEX bit in the MACTCTL register. Note that in 10BASE-T half-duplex mode, the transmitted data is looped back on the receive path.

The Ethernet MAC automatically generates and inserts the Frame Check Sequence (FCS) at the end of the transmit frame when the CRC bit in the MACTCTL register is set. However, for test purposes, this feature can be disabled in order to generate a frame with an invalid CRC by clearing the CRC bit.

The IEEE 802.3 specification requires that the Ethernet frame payload section be a minimum of 46 bytes. The Ethernet MAC automatically pads the data section if the payload data section loaded into the FIFO is less than the minimum 46 bytes when the PADEN bit in the MACTCTL register is set. This feature can be disabled by clearing the PADEN bit.

The transmitter must be enabled by setting the TXEN bit in the MACTCTL register.

19.3.1.4 Ethernet Reception Options

The Ethernet MAC RX FIFO should be cleared during software initialization. The receiver should first be disabled by clearing the RXEN bit in the Ethernet MAC Receive Control (MACRCTL) register, then the FIFO can be cleared by setting the RSTFIFO bit in the MACRCTL register.

Copyright © 2012–2019, Texas Instruments Incorporated

The receiver automatically rejects frames that contain bad CRC values in the FCS field. In this case, a Receive Error interrupt is generated and the receive data is lost. To accept all frames, clear the BADCRC bit in the MACRCTL register.

In normal operating mode, the receiver accepts only those frames that have a destination address that matches the address programmed into the Ethernet MAC Individual Address 0 (MACIA0) and Ethernet MAC Individual Address 1 (MACIA1) registers. However, the Ethernet receiver can also be configured for Promiscuous and Multicast modes by setting the PRMS and AMUL bits in the MACRCTL register.

19.3.1.5 Packet Timestamps

Some applications require a very precise clock for time stamping samples or triggering events. The IEEE Precision Time Protocol (PTP), or IEEE-1588, provides a mechanism for synchronizing clocks across an Ethernet to sub-microsecond precision. The accuracy of the PTP clock depends greatly upon the accuracy of timestamps of the PTP Ethernet packets. In a software-only PTP solution, there can be jitter in the Ethernet packet timestamps, resulting in a less precise PTP clock on the target. In some devices, General-Purpose Timer 3 (GPT3) can be used in conjunction with the Ethernet MAC Timer Support (MACTS) register to provide a more accurate timestamp for Ethernet packets.

This feature is enabled by setting the TSEN bit in the MACTS register. Note that when this feature is enabled, GPT3 must be dedicated to the Ethernet MAC. GPT3 must be configured to 16-bit edge capture mode. This is described in the *GP Timers* chapter. Timer A of GPT3 stores the transmit time, and Timer B stores the receive time. One other General-Purpose Timer can be set up as a 16-bit free-running timer to synchronize the receiver and transmitter timers and provide a timestamp with which to compare the timestamps stored in GPT3. The enet_ptpd example in the controlSUITE™ software package provides a sample PTP application that illustrates both software-only time stamping as well the use of the GPT3 and MACTS register for more accurate timestamps. This example supports version 1 of the IEEE-1588 protocol, but Concerto™ microcontrollers support both versions 1 and 2.

19.3.1.6 Ethernet MAC Address

The Ethernet MAC address consists of 6 octets that uniquely identify the Ethernet MAC. The MAC address is split up into EMACID0, which should be stored at OTP location 0x680810 and EMACID1, which should be stored at OTP location 0x680814. See the example below for how to store the MAC address into OTP.

MAC Address of 12-34-56-78-9A-BC should be stored as:

- EMACID0: 0x00563412 (stored at 0x681010)
- EMACID1: 0x00BC9A78 (stored at 0x681014)

  The top byte of each EMACID location is 0x00. Only the lower 3 bytes of each EMACID location are used as shown above.

19.3.2 Internal MII Operation

For the MII management interface to function properly, the internal clock must be divided down from the system clock to a frequency no greater than 2.5 MHz. The Ethernet MAC Management Divider (MACMDV) register contains the divider used for scaling down the system clock.

19.3.3 Interrupts

The Ethernet MAC can generate an interrupt for one or more of the following conditions:

- A frame has been received into an empty RX FIFO
- A frame transmission error has occurred
- A frame has been transmitted successfully
- A frame has been received with inadequate room in the RX FIFO (overrun)
- A frame has been received with one or more error conditions (for example, FCS failed)
- An MII management transaction between the MAC and PHY layers has completed

Copyright © 2012–2019, Texas Instruments Incorporated

19.3.4 DMA Operation

The Ethernet peripheral provides request signals to the μDMA controller and has a dedicated channel for transmit and one for receive. The request is a single type for both channels. Burst requests are not supported. The RX channel request is asserted when a packet is received while the TX channel request is asserted when the transmit FIFO becomes empty.

No special configuration is needed to enable the ethernet peripheral for use with the μDMA controller.

Because the size of a received packet is not known until the header is examined, it is best to set up the initial μDMA transfer to copy the first 4 words including the packet length plus the Ethernet header from the RX FIFO when the RX request occurs. The μDMA causes an interrupt when this transfer is complete. Upon entering the interrupt handler, the packet length in the FIFO and the Ethernet header are in a buffer and can be examined. Once the packet length is known, then another μDMA transfer can be set up to transfer the remaining received packet payload from the FIFO into a buffer. This transfer should be initiated by software. Another interrupt occurs when this transfer is done.

Even though the TX channel generates a TX empty request, the recommended way to handle μDMA transfers for transmitting packets is to set up the transfer from the buffer containing the packet to the transmit FIFO, and then to initiate the transfer with a software request. An interrupt occurs when this transfer is complete. For both channels, the "auto-request" transfer mode should be used. See the *Micro Direct Memory Access* (μDMA) chapter for more details about programming the μDMA controller.

19.4 Initialization and Configuration

The following sections describe the hardware and software configuration required to set up the Ethernet MAC.

19.4.1 Software Configuration

To use the Ethernet MAC, it must be enabled by setting the EMAC0 bit in the RCGC2 register. In addition, the clock to the appropriate GPIO module must be enabled via the RCGC2 register in the System Control module. To find out which GPIO port to enable, refer to the GPIO chapter.

The following steps can then be used to configure the Ethernet MAC for basic operation.

1. 1. Program the MACDIV register to obtain a 2.5 MHz clock (or less) on the internal MII. Assuming a 100-MHz system clock, the MACDIV value should be 0x13 or greater.
2. Program the MACIA0 and MACIA1 register for address filtering.
3. Program the MACTCTL register for Auto CRC generation, padding, and full-duplex operation using a value of 0x16.
4. Program the MACRCTL register to flush the receive FIFO and reject frames with bad FCS using a value of 0x18.
5. Enable both the Transmitter and Receive by setting the LSB in both the MACTCTL and MACRCTL registers.
6. To transmit a frame, write the frame into the TX FIFO using the Ethernet MAC Data (MACDATA) register. Then set the NEWTX bit in the Ethernet Mac Transmission Request (MACTR) register to initiate the transmit process. When the NEWTX bit has been cleared, the TX FIFO is available for the next transmit frame.
7. To receive a frame, wait for the NPR field in the Ethernet MAC Number of Packets (MACNP) register to be non-zero. Then begin reading the frame from the RX FIFO by using the MACDATA register. To ensure that the entire packet is received, either use the DriverLib EthernetPacketGet() API or compare the number of bytes received to the Length field from the frame to determine when the packet has been completely read.

Copyright © 2012–2019, Texas Instruments Incorporated

19.5 Register Map

Table 19-2 lists the EMAC and MII Management registers. The MAC register addresses given are relative to the Ethernet base address of 0x4004.8000. The MII Management registers are accessed using the MACMCTL register. Note that the EMAC clocks must be enabled before the registers can be programmed. There must be a delay of three system clocks after the Ethernet MAC clock is enabled before any MAC registers are accessed.

The IEEE 802.3 standard specifies a register set for controlling and gathering status from the PHY layer. The registers are collectively known as the MII Management registers and are detailed in Section 22.2.4 of the IEEE 802.3 specification. Section 19.7 also lists these MII Management registers. All addresses given are absolute and are written directly to the REGADR field of the Ethernet MAC Management Control (MACMCTL) register. The format of registers 0 to 15 are defined by the IEEE specification and are common to all PHY layer implementations. The only variance allowed is for features that may or may not be supported by a specific PHY implementation. Registers 16 to 31 are vendor-specific registers, used to support features that are specific to a vendor's PHY implementation.

PHY registers MR0 – MR6 are not located on the microcontroller. These registers are located on IEEE 802.3 compliant Ethernet external PHYs. These registers are defined for software ease of use. See registers MACMCTL, MACMTXD, and MACMRXD for instructions on how to read and write to the registers on an external Ethernet PHY.

Table 19-2. Ethernet Register Map

Offset	Name	Type	Reset	Description
0x000	MACRIS/MACIACK	R/W1C	0x0000.0000	Ethernet MAC Raw Interrupt Status/Acknowledge
0x004	MACIM	R/W	0x0000.007F	Ethernet MAC Interrupt Mask
0x008	MACRCTL	R/W	0x0000.0008	Ethernet MAC Receive Control
0x00C	MACTCTL	R/W	0x0000.0000	Ethernet MAC Transmit Control
0x010	MACDATA	R/W	0x0000.0000	Ethernet MAC Data
0x014	MACIA0	R/W	0x0000.0000	Ethernet MAC Individual Address 0
0x018	MACIA1	R/W	0x0000.0000	Ethernet MAC Individual Address 1
0x01C	MACTHR	R/W	0x0000.003F	Ethernet MAC Threshold
0x020	MACMCTL	R/W	0x0000.0000	Ethernet MAC Management Control
0x024	MACMDV	R/W	0x0000.0080	Ethernet MAC Management Divider
0x028	MACMAR	R/W	0X0	Ethernet MAC Management Address Register
0x02C	MACMTXD	R/W	0x0000.0000	Ethernet MAC Management Transmit Data
0x030	MACMRXD	R/W	0x0000.0000	Ethernet MAC Management Receive Data
0x034	MACNP	RO	0x0000.0000	Ethernet MAC Number of Packets
0x038	MACTR	R/W	0x0000.0000	Ethernet MAC Transmission Request
0x03C	MACTS	R/W	0x0000.0000	Ethernet MAC Timer Support
0x040	-	R/W	0x0000.0000	Reserved
0x044	-	R/W	0x0000.0000	Reserved
MII Management (Accessed through the MACMCTL register)				
-	MR0	R/W	0x1000	Ethernet PHY Management Register 0 – Control
-	MR1	RO	0x7809	Ethernet PHY Management Register 1 – Status

Copyright © 2012–2019, Texas Instruments Incorporated

Table 19-2. Ethernet Register Map (continued)

Offset	Name	Type	Reset	Description
-	MR2	RO	0x0161	Ethernet PHY Management Register 2 – PHY Identifier 1
-	MR3	RO	0xB410	Ethernet PHY Management Register 3 – PHY Identifier 2
-	MR4	R/W	0x01E1	Ethernet PHYManagement Register 4 – Auto-Negotiation Advertisement
-	MR5	RO	0x0001	Ethernet PHYManagement Register 5 – Auto-Negotiation Link Partner Base Page Ability
-	MR6	RO	0x0000	Ethernet PHYManagement Register 6 – Auto-Negotiation Expansion

19.6 Ethernet MAC Register Descriptions

The remainder of this section lists lists and describes the Ethernet MAC registers, in numerical order by address offset. Also see Section 19.7.

19.6.1 Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK) Register, offset 0x000

The MACRIS/MACIACK register is the interrupt status and acknowledge register. On a read, this register gives the current status value of the corresponding interrupt prior to masking. On a write, setting any bit clears the corresponding interrupt status bit.

Figure 19-4. Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK) Register

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved	PHYINT	MDINT	RXER	FOV	TXEMP	TXER	RXINT
R-0	R/W-1C	R/W-1C	R/W-1C	R/W-1C	R/W-1C	R/W-1C	R/W-1C

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-3. Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK) Register Field Descriptions

Bit	Field	Value	Description
31-7	Reserved		Reserved
6	PHYINT		PHY Interrupt
		0	No interrupt.
		1	An enabled interrupt in the PHY layer has occurred. Check the appropriate PHY register to determine the specific PHY event that triggered this interrupt.
			This bit is cleared by writing a 1 to it.
5	MDINT		MII Transaction Complete
		0	No interrupt.
		1	A transaction (read or write) on the MII interface has completed successfully.
			This bit is cleared by writing a 1 to it.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 19-3. Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK) Register Field Descriptions (continued)

Bit	Field	Value	Description
4	RXER		Receive Error
		0	No interrupt
		1	An error was encountered on the receiver. The possible errors that can cause this interrupt bit to be set are: 1 • A receive error occurs during the reception of a frame (100 Mbps only). • The frame is not an integer number of bytes (dribble bits) due to an alignment error. • The CRC of the frame does not pass the FCS check. • The length/type field is inconsistent with the frame data size when interpreted as a length field. This bit is cleared by writing a 1 to it.
3	FOV		FIFO Overrun
		0	No interrupt
		1	An overrun was encountered on the receive FIFO.
2	TXEMP		Transmit FIFO Empty
		0	No interrupt
		1	The packet was transmitted and that the TX FIFO is empty. This bit is cleared by writing a 1 to it.
1	TXER		Transmit Error
		0	No interrupt
		1	An error was encountered on the transmitter. The possible errors that can cause this interrupt bit to be set are: 1 • The data length field stored in the TX FIFO exceeds 2032 decimal (buffer length - 16 bytes of header data). The frame is not sent when this error occurs. • The retransmission attempts during the backoff process have exceeded the maximum limit of 16 decimal. Writing a 1 to this bit clears it and resets the TX FIFO write pointer.
0	RXINT		Packet Received
		0	No interrupt.
		1	At least one packet has been received and is stored in the receiver FIFO This bit is cleared by writing a 1 to it.

19.6.2 Ethernet MAC Interrupt Mask (MACIM) Register, offset 0x004

The Ethernet MAC Interrupt Mask (MACIM) register is shown and described in the figure and table below.

Figure 19-5. Ethernet MAC Interrupt Mask (MACIM) Register

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

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved	PHYINTM	MDINTM	RXERM	FOVM	TXEMPM	TXERM	RXINTM
R-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1C	R/W-1

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 19-4. Ethernet MAC Interrupt Mask (MACIM) Register Field Descriptions

Bit	Field	Value	Description
31-7	Reserved		Reserved
6	PHYINT		Mask PHY Interrupt
		0	The PHYINT interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the PHYINT bit in the MACRIS/MACIACK register is set.
5	MDINTM		Mask MII Transaction Complete
		0	The MDINT interrupt is suppressed and not sent to the interrupt controller
		1	An interrupt is sent to the interrupt controller when the MDINT bit in the MACRIS/MACIACK register is set.
4	RXERM		Mask Receive Error
		0	The RXER interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the RXER bit in the MACRIS/MACIACK register is set
3	FOVM		Mask FIFO Overrun
		0	The FOV interrupt is suppressed and not sent to the interrupt controller
		1	An interrupt is sent to the interrupt controller when the FOV bit in the MACRIS/MACIACK register is set.
2	TXEMPM		Mask Transmit FIFO Empty
		0	The TXEMP interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the TXEMP bit in the MACRIS/MACIACK register is set.
1	TXERM		Mask Transmit Error
		0	The TXER interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the TXER bit in the MACRIS/MACIACK register is set.
0	RXINTM		Mask Packet Received
		0	The RXINT interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the RXINT bit in the MACRIS/MACIACK register is set.

19.6.3 Ethernet MAC Receive Control (MACRCTL) Register, offset 0x008

The Ethernet MAC Receive Control (MACRCTL) register is shown and described in the figure and table below.

Figure 19-6. Ethernet MAC Receive Control (MACRCTL) Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											RSTFIFO	BADCRC	PRMS	AMUL	RXEN
R-0											R/W-0	R/W-1	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 19-5. Ethernet MAC Receive Control (MACRCTL) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4	RSTFIFO		Clear Receive FIFO
		0	No effect.
		1	Clear the receive FIFO. The receive FIFO should be cleared when software initialization is performed. This bit is automatically cleared when read. The receiver should be disabled (RXEN = 0), before a reset is initiated (RSTFIFO = 1). This sequence flushes and resets the RX FIFO.
3	BADCRC		Enable Reject Bad CRC
		0	Disables the rejection of frames with an incorrectly calculated CRC.
		1	Enables the rejection of frames with an incorrectly calculated CRC. If a bad CRC is encountered, the RXER bit in the MACRIS register is set and the receiver FIFO is reset.
2	PRMS		Enable Promiscuous Mode
		0	Disables Promiscuous mode, accepting only frames with the programmed Destination Address.
		1	Enables Promiscuous mode, which accepts all valid frames, regardless of the specified Destination Address.
1	AMUL		Enable Multicast Frames
		0	Disables the reception of multicast frames.
		1	Enables the reception of multicast frames.
0	RXEN		Enable Receiver
		0	Disables the receiver. All frames are ignored.
		1	Enables the Ethernet receiver

19.6.4 Ethernet MAC Transmit Control (MACTCTL) Register, offset 0x00C

The Ethernet MAC Transmit Control (MACTCTL) register is shown and described in the figure and table below.

Figure 19-7. Ethernet MAC Transmit Control (MACTCTL) Register

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		DUPLEX	Reserved	CRC	PADEN	TXEN
R-0		R/W-0	R-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-6. Ethernet MAC Transmit Control (MACTCTL) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4	DUPLEX		Enable duplex mode
		0	Disables duplex mode.
		1	Enables duplex mode, allowing simultaneous transmission and reception.
3	Reserved		Reserved
2	CRC		Enable CRC Generation
		0	The frames placed in the TX FIFO are sent exactly as they are written into the FIFO.
		1	Enables the automatic generation of the CRC and its placement at the end of the packet. Note that this bit should generally be set.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 19-6. Ethernet MAC Transmit Control (MACTCTL) Register Field Descriptions (continued)

Bit	Field	Value	Description
1	PADEN		Enable packet padding
		0	Disables automatic padding.
		1	Enables the automatic padding of packets that do not meet the minimum frame size. Note that this bit should generally be set.
0	TXEN		Enable transmitter
		0	Disables the transmitter
		1	Enables the transmitter

Copyright © 2012–2019, Texas Instruments Incorporated

19.6.5 Ethernet MAC Data (MACDATA) Register, offset 0x010

The Ethernet MAC Data (MACDATA) register enables software to access the TX and RX FIFOs. Reads from this register return the data stored in the RX FIFO from the location indicated by the read pointer. The read pointer is then auto incremented to the next RX FIFO location. Reading from the RX FIFO when a frame has not been received or is in the process of being received returns indeterminate data and does not increment the read pointer.

Writes to this register store the data in the TX FIFO at the location indicated by the write pointer. The write pointer is then auto incremented to the next TX FIFO location. Writing more data into the TX FIFO than indicated in the length field results in the data being lost. Writing less data into the TX FIFO than indicated in the length field results in indeterminate data being appended to the end of the frame to achieve the indicated length. Attempting to write the next frame into the TX FIFO before transmission of the first has completed results in the data being lost.

Bytes may not be randomly accessed in either the RX or TX FIFOs. Data must be read from the RX FIFO sequentially and stored in a buffer for further processing. Once a read has been performed, the data in the FIFO cannot be re-read. Data must be written to the TX FIFO sequentially. If an error is made in placing the frame into the TX FIFO, the write pointer can be reset to the start of the TX FIFO by writing the TXER bit of the MACIACK register and then the data re-written.

READS

Figure 19-8. Ethernet MAC Data (MACDATA) Register (READ)

31	0
RXDATA	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-7. Ethernet MAC Data (MACDATA) Register (READ) Field Descriptions

Bit	Field	Value	Description
31-0	RXDATA	0x0000.0000	Receive FIFO Data The RXDATA bits represent the next word of data stored in the RX FIFO.

WRITES

Figure 19-9. Ethernet MAC Data (MACDATA) Register (WRITE)

31	0
TXDATA	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-8. Ethernet MAC Data (MACDATA) Register (WRITE) Field Descriptions

Bit	Field	Value	Description
31-0	TXDATA	0x0000.0000	Transmit FIFO Data The TXDATA bits represent the next word of data stored in the TX FIFO for transmission.

Copyright © 2012–2019, Texas Instruments Incorporated

19.6.6 Ethernet MAC Individual Address 0 (MACIA0) Register, offset 0x014

The Ethernet MAC Individual Address 0 (MACIA0) register enables software to program the first four bytes of the hardware MAC address of the Network Interface Card (NIC). The last two bytes are in MACIA1. The 6-byte Individual Address is compared against the incoming Destination Address fields to determine whether the frame should be received

Figure 19-10. Ethernet MAC Individual Address 0 (MACIA0) Register

31	24	23	16
MACOCT4		MACOCT3	
R/W-0		R/W-0	

15	8	7	0
MACOCT2		MACOCT1	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-9. Ethernet MAC Individual Address 0 (MACIA0) Register Field Descriptions

Bit	Field	Value	Description
31-24	MACOCT4	00h	MAC Address Octet 4 The MACOCT4 bits represent the fourth octet of the MAC address used to uniquely identify the Ethernet MAC.
23-16	MACOCT3	00h	MAC Address Octet 3 The MACOCT3 bits represent the fourth octet of the MAC address used to uniquely identify the Ethernet MAC.
15-8	MACOCT2	00h	MAC Address Octet 2 The MACOCT2 bits represent the fourth octet of the MAC address used to uniquely identify the Ethernet MAC.
7-0	MACOCT1	00h	MAC Address Octet 1 The MACOCT1 bits represent the fourth octet of the MAC address used to uniquely identify the Ethernet MAC.

Copyright © 2012–2019, Texas Instruments Incorporated

19.6.7 Ethernet MAC Individual Address 1 (MACIA1) Register, offset 0x018

The Ethernet MAC Individual Address 1 (MACIA1) register enables software to program the last two bytes of the hardware MAC address of the Network Interface Card (NIC). The first four bytes are in MACIA0. The 6-byte IAR is compared against the incoming Destination Address fields to determine whether the frame should be received.

Figure 19-11. Ethernet MAC Individual Address 0 (MACIA1) Register

31	16
Reserved	
R-0	

15	8	7	0
MACOCT6		MACOCT5	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-10. Ethernet MAC Individual Address 0 (MACIA1) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-8	MACOCT6	00h	MAC Address Octet 6 The MACOCT6 bits represent the sixth octet of the MAC address used to uniquely identify each Ethernet MAC.
7-0	MACOCT5	00h	MAC Address Octet 5 The MACOCT5 bits represent the fifth octet of the MAC address used to uniquely identify the Ethernet MAC.

Copyright © 2012–2019, Texas Instruments Incorporated

19.6.8 Ethernet MAC Threshold (MACTHR) Register, offset 0x01C

In order to increase the transmission rate, it is possible to program the Ethernet MAC to begin transmission of the next frame prior to the completion of the transmission of the current frame. Caution – Extreme care must be used when implementing this function. Software must be able to guarantee that the complete frame is able to be stored in the transmission FIFO prior to the completion of the transmission frame. This register enables software to set the threshold level at which the transmission of the frame begins. If the THRESH bits are set to 0x3F, which is the reset value, the early transmission feature is disabled, and transmission does not start until the NEWTX bit is set in the MACTR register.

Writing the THRESH field to any value besides 0x3F enables the early transmission feature. Once the byte count of data in the TX FIFO reaches the value derived from the THRESH bits as shown below, transmission of the frame begins. When the THRESH field is clear, transmission of the frame begins after 4 bytes (a single write) are stored in the TX FIFO. Each increment of the THRESH bit field waits for an additional 32 bytes of data (eight writes) to be stored in the TX FIFO. Therefore, a value of 0x01 causes the transmitter to wait for 36 bytes of data to be written while a value of 0x02 makes the wait equal to 68 bytes of written data. In general, early transmission starts when:

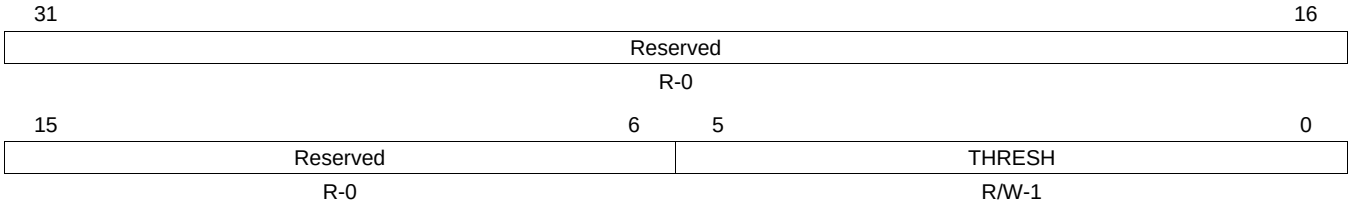
Number of Bytes ≥ 4 ((THRESH x 8) + 1)

Reaching the threshold level has the same effect as setting the NEWTX bit in the MACTR register. Transmission of the frame begins, and then the number of bytes indicated by the Data Length field is transmitted. Because underrun checking is not performed, if any event, such as an interrupt, delays the filling of the FIFO, the tail pointer may reach and pass the write pointer in the TX FIFO. In this event, indeterminate values are transmitted rather than the end of the frame. Therefore, sufficient bus bandwidth for writing to the TX FIFO must be guaranteed by the software.

If a frame smaller than the threshold level must be sent, the NEWTX bit in the MACTR register must be set with an explicit write, which initiates the transmission of the frame even though the threshold limit has not been reached.

If the threshold level is set too small, it is possible for the transmitter to underrun. If this occurs, the transmit frame is aborted, and a transmit error occurs. Note that in this case, the TXER bit in the MACRIS is not set, meaning that the CPU receives no indication that a transmit error happened.

Figure 19-12. Ethernet MAC Threshold (MACTHR) Register



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 19-11. Ethernet MAC Threshold (MACTHR) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-0	THRESH		Threshold Value
		3Fh	The THRESH bits represent the early transmit threshold. Once the amount of data in the TX FIFO exceeds the value represented by the above equation, transmission of the packet begins.

Copyright © 2012–2019, Texas Instruments Incorporated

19.6.9 Ethernet MAC Management Control (MACMCTL) Register, offset 0x020

The Ethernet MAC Management Control (MACMCTL) register enables software to control the transfer of data to and from the MII Management registers in an external Ethernet PHY. The address, name, type, reset configuration, and functional description of each of these registers can be found in Section 19.7.

In order to initiate a read transaction from the MII Management registers, the WRITE bit must be cleared during the same cycle that the START bit is set. In order to initiate a write transaction to the MII Management registers, the WRITE bit must be set during the same cycle that the START bit is set.

Figure 19-13. Ethernet MAC Management Control (MACMCTL) Register

31	8	7	3	2	1	0
Reserved		REGADR		Reserved	WRITE	START
R-0		R/W-0		R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-12. Ethernet MAC Management Control (MACMCTL) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-3	REGADR		MII Register Address
		0h	The REGADR bit field represents the MII Management register address for the next MII management interface transaction. Refer to Table 19-2 for the PHY register offsets. Note that any address that is not valid in the register map should not be written to, and any data read should be ignored.
2	Reserved		Reserved
1	WRITE		MII Register Transaction Type
		0	The next operation of the next MII management interface is a read transaction
		1	The next operation of the next MII management interface is a write transaction
0	START		MII Register Transaction Enable
		0	No effect.
		1	The MII register located at REGADR is read (WRITE=0) or written (WRITE=1).

Copyright © 2012–2019, Texas Instruments Incorporated

19.6.10 Ethernet MAC Management Divider (MACMDV) Register, offset 0x024

The Ethernet MAC Management Divider (MACMDV) register enables software to set the clock divider for the Management Data Clock (MDC). This clock is used to synchronize read and write transactions between the system and the MII Management registers. The frequency of the MDC clock can be calculated from the following formula:

$$F_{mdc} = \frac{F_{ipclk}}{2 \times (\text{MACMDV} + 1)}$$

The clock divider must be written with a value that ensures that the MDC clock does not exceed a frequency of 2.5 MHz.

Figure 19-14. Ethernet MAC Management Divider (MACMDV) Register

31	8	7	0
Reserved		DIV	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-13. Ethernet MAC Management Divider (MACMDV) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	DIV		Clock divider
		80h	The DIV bits are used to set the clock divider for the MDC clock used to transmit data between the MAC and external PHY layers.

19.6.11 Ethernet MAC Management Address Register (MACMAR), offset 0x028

The Ethernet MAC Management Address register (MACMAR) allows software to choose the address of the PHY from the next MII management register transaction. Because there is currently only a single PHY in Fury, the PHYADR bits should not be written and left at 0x00.

Figure 19-15. Ethernet MAC Management Address Register (MACMAR)

31	5	4	0
Reserved		PHYADR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-14. Ethernet MAC Management Address Register (MACMAR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4-0	PHYADR	0000h	The PHYADR bits represent the address of the PHY that will be accessed in the next MII management transaction.

19.6.12 Ethernet MAC Management Transmit Data (MACMTXD) Register, offset 0x02C

The Ethernet MAC Management Transmit Data (MACMTXD) register holds the next value to be written to the MII Management registers.

Figure 19-16. Ethernet MAC Management Transmit Data (MACMTXD) Register

31	16	15	0
Reserved		MDTX	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 19-15. Ethernet MAC Management Transmit Data (MACMTXD) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	MDTX		MII Register Transmit Data
		0000h	The MDTX bits represent the data to be written in the next MII management transaction

Copyright © 2012–2019, Texas Instruments Incorporated

19.6.13 Ethernet MAC Management Receive Data (MACMRXD) Register, offset 0x030

The Ethernet MAC Management Receive Data (MACMRXD) register holds the last value read from the MII Management registers.

Figure 19-17. Ethernet MAC Management Receive Data (MACMRXD) Register

31	16	15	0
Reserved		MDRX	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-16. Ethernet MAC Management Receive Data (MACMRXD) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	MDRX		MII Register Recieve Data
		0000h	The MDRX bits represent the data to be written in the previous MII management transaction

19.6.14 Ethernet MAC Number of Packets (MACNP) Register, offset 0x034

The Ethernet MAC Number of Packets (MACNP) register holds the number of frames that are currently in the RX FIFO. When NPR is 0, there are no frames in the RX FIFO, and the RXINT bit is clear. When NPR is any other value, at least one frame is in the RX FIFO, and the RXINT bit in the MACRIS register is set.

NOTE: The FCS bytes are not included in the NPR value. As a result, the NPR value could be zero before the FCS bytes are read from the FIFO. In addition, a new packet could be received before the NPR value reaches zero. To ensure that the entire packet is received, either use the DriverLib EthernetPacketGet() API or compare the number of bytes received to the Length field from the frame to determine when the packet has been completely read.

Figure 19-18. Ethernet MAC Number of Packets (MACNP) Register

31	6	5	0
Reserved		NPR	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-17. Ethernet MAC Number of Packets (MACNP) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-0	NPR		Number of Packets in Receive FIFO
		00h	The NPR bits represent the number of packets stored in the RX FIFO. While the NPR field is greater than 0, the RXINT interrupt in the MACRIS register is set.

Copyright © 2012–2019, Texas Instruments Incorporated

19.6.15 Ethernet MAC Transmission Request (MACTR) Register, offset 0x038

The Ethernet MAC Transmission Request (MACTR) register enables software to initiate the transmission of the frame currently located in the TX FIFO. Once the frame has been transmitted from the TX FIFO or a transmission error has been encountered, the NEWTX bit is automatically cleared.

Figure 19-19. Ethernet MAC Transmission Request (MACTR) Register

31	1	0
Reserved		NEWTX
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-18. Ethernet MAC Transmission Request (MACTR) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	NEWTX		New Transmission
		0	The transmission has completed.
		1	Initiates an Ethernet transmission once the packet has been placed in the TX FIFO If early transmission is being used (see the MACTHR register), this bit does not need to be set.

19.6.16 Ethernet MAC Timer Support (MACTS) Register, offset 0x03C

The Ethernet MAC Timer Support (MACTS) register enables software to enable highly precise timing on the transmission and reception of frames. To enable this function, set the TSEN bit.

Figure 19-20. Ethernet MAC Timer Support (MACTS) Register

31	1	0
Reserved		TSEN
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-19. Ethernet MAC Timer Support (MACTS) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	TSEN		Time Stamp Enable
		0	No effect.
		1	The TX and RX interrupts are routed to the CCP inputs of General-Purpose Timer 3.

Copyright © 2012–2019, Texas Instruments Incorporated

19.7 MII Management Register Descriptions

The IEEE 802.3 standard specifies a register set for controlling and gathering status from the PHY layer. The registers are collectively known as the MII Management registers. The Section 19.6.9 is used to access the MII Management registers. All addresses given are absolute. Addresses not listed are reserved; these addresses should not be written to and any data read should be ignored. Also see Table 19-2.

PHY registers MR0 – MR6 are not located on the microcontroller. These registers are located on IEEE 802.3 compliant Ethernet external PHYs. These registers are defined for software ease of use. See registers MACMCTL, MACMTXD, and MACMRXD for instructions on how to read and write to the registers on an external Ethernet PHY.

19.7.1 Ethernet PHY Management Register 0 – Control (MR0) Register, address 0x00

This register enables software to configure the operation of an external PHY. The default settings of these registers are designed to initialize the Ethernet external PHY to a normal operational mode without configuration

Figure 19-21. Ethernet PHY Management Register 0 – Control (MR0) Register

15	14	13	12	11	10	9	8
RESET	LOOPBK	SPEEDSL	ANEGEN	PWRDN	ISO	RANEG	DUPLEX
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	0
COLT	Reserved	
R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-20. Ethernet PHY Management Register 0 – Control (MR0) Register Field Descriptions

Bit	Field	Value	Description
15	RESET		Reset Registers
		0	No effect
		1	The PHY layer registers reset to their default state and the internal state machines are reinitialized
			Once the reset operation has completed, this bit is automatically cleared by hardware.
14	LOOPBK		Loopback Mode
		0	No effect
		1	Enables the Loopback mode of operation. The receiver ignores external inputs and receives the data that is transmitted by the transmitter.
13	SPEEDSL		Speed Select
		0	Enables the 10 Mbps mode of operation (10BASE-T).
		1	Enables the 100 Mbps mode of operation (100BASE-TX).
12	ANEGEN		Auto-Negotiation Enable
		0	No effect
		1	Enables the auto-negotiation process.
11	PWRDN		Power down
		0	No effect
		1	The PHY layer is configured to be in a low-power consuming state. All data on the data inputs is ignored.
10	ISO		Isolate
		0	No effect
		1	The transmit and receive data paths are isolated and all data being transmitted and received is ignored.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 19-20. Ethernet PHY Management Register 0 – Control (MR0) Register Field Descriptions (continued)

Bit	Field	Value	Description
9	RANEG		Restart Auto-Negotiation
		0	No effect
		1	Restarts the auto-negotiation process.
8	DUPLEX		Set Duplex Mode
		0	Enables the Half-Duplex mode of operation. Note that in 10BASE-T half-duplex mode, the transmitted data is looped back on the receive path.
		1	Enables the Full-Duplex mode of operation. This bit can be set by software in a manual configuration process or by the auto-negotiation process.
7	COLT		Collision Test
		0	No effect
		1	Enables the Collision Test mode of operation.
6-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

19.7.2 Ethernet PHY Management Register 1 – Status (MR1) Register, address 0x01

This register enables software to determine the capabilities of an external PHY and perform its initialization and operation appropriately.

Figure 19-22. Ethernet PHY Management Register 1 – Status (MR1) Register

15	14	13	12	11	10		8
Reserved	100X_F	100X_H	10T_F	10T_H	Reserved		
R-0	R-1	R-1	R-1	R-1	R-0		
7	6	5	4	3	2	1	0
Reserved		ANEGC	RFAULT	ANEGA	LINK	JAB	EXTD
R-0		R-0	R-0	R-0	R-0	R-C	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 19-21. Ethernet PHY Management Register 1 – Control (MR1) Register Field Descriptions

Bit	Field	Value	Description
15	Reserved	1	Reserved
14	100X_F		100BASE-TX Full-Duplex Mode
		0	The Ethernet PHY is not capable of supporting 100BASE-TX Full-Duplex mode.
		1	The Ethernet PHY is capable of supporting 100BASE-TX Full-Duplex mode.
13	100X_H		100BASE-TX Half-Duplex Mode
		0	The Ethernet PHY is not capable of supporting 100BASE-TX Half-Duplex mode
		1	The Ethernet PHY is capable of supporting 100BASE-TX Half-Duplex mode.
12	10T_F		10BASE-T Full-Duplex Mode
		0	The Ethernet PHY is not capable of supporting 10BASE-T Full-Duplex mode.
		1	The Ethernet PHY is capable of supporting 10BASE-T Full-Duplex mode.
11	10T_H		10BASE-T Half-Duplex Mode
		0	The Ethernet PHY is not capable of supporting 10BASE-T Half-Duplex mode.
		1	The Ethernet PHY is capable of supporting 10BASE-T Half-Duplex mode.
10-6	Reserved		Reserved
5	ANEGC		Auto-Negotiation Complete
		0	The auto-negotiation process is not complete.
		1	The auto-negotiation process has been completed and that the extended registers defined by the auto-negotiation protocol are valid.
4	RFAULT		Remote Fault
		0	A remote fault condition has not been detected.
		1	A remote fault condition has been detected. This bit remains set until it is read, even if the condition no longer exists.
3	ANEGA		Auto-Negotiation
		0	The Ethernet PHY does not have the ability to perform auto-negotiation.
		1	The Ethernet PHY has the ability to perform auto-negotiation.
2	LINK		Link Made
		0	A valid link has not been established by the Ethernet PHY.
		1	A valid link has been established by the Ethernet PHY.
1	JAB		Jabber Condition
		0	A jabber condition has not been detected by the Ethernet PHY.
		1	A jabber condition has been detected by the Ethernet Controller This bit remains set until it is read, even if the jabber condition no longer exists.
0	EXTD		Extended Capabilities
		0	The Ethernet PHY does not provide extended capabilities
		1	The Ethernet PHY provides an extended set of capabilities that can be accessed through the extended register set.

Copyright © 2012–2019, Texas Instruments Incorporated

19.7.3 Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2) Register , address 0x02

The Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2) register, along with MR3, provides a 32-bit value indicating the manufacturer, model, and revision information.

Figure 19-23. Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2) Register

15	0
OUI[21:6]	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-22. Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2) Register Field Descriptions

Bit	Field	Value	Description
15-0	OUI[21:6]	161h	Organizationally Unique Identifier[21:6] This field, along with the OUI[5:0] field in MR3, makes up the Organizationally Unique Identifier indicating the PHY manufacturer.

19.7.4 Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3) Register, address 0x03

The Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3) register, along with MR2, provides a 32-bit value indicating the manufacturer, model, and revision information.

Figure 19-24. Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3) Register

15	10	9	4	3	0
OUI[5:0]		MN		RN	
		R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-23. Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3) Register Field Descriptions

Bit	Field	Value	Description
15-10	OUI[5:0]	2Dh	Organizationally Unique Identifier[5:0] This field, along with the OUI[21:6] field in MR2, makes up the Organizationally Unique Identifier indicating the PHY manufacturer.
9-4	MN	01h	Model Number The MN field represents the Model Number of the PHY.
3-0	RN	0h	Revision Number The RN field represents the Revision Number of the PHY implementation.

Copyright © 2012–2019, Texas Instruments Incorporated

19.7.5 Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4) Register, address 0x04

The Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4) register provides the advertised abilities of the Ethernet PHY used during auto-negotiation. Bits 8:5 represent the Technology Ability Field bits. This field can be overwritten by software to auto-negotiate to an alternate common technology. Writing to this register has no effect until auto-negotiation is re-initiated by setting the RANEG bit in the MR0 register.

Figure 19-25. Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4) Register

15	14	13	12	9	8
NP	Reserved	RF	Reserved		A3
R-0	R-0	R/W-0	R-0		R/W-1

7	6	5	4	0
A2	A1	A0	S	
R/W-1	R/W-1	R/W-1	R-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-24. Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4) Register Field Descriptions

Bit	Field	Value	Description
15	NP		Next Page
		0	The Ethernet PHY is not capable of Next Page exchanges.
		1	The Ethernet PHY is capable of Next Page exchanges to provide more detailed information on the PHY layer's capabilities.
14	Reserved		Reserved
13	RF		Remote Fault
		0	No Remote Fault condition has been encountered.
		1	Indicates to the link partner that a Remote Fault condition has been encountered.
12-9	Reserved		Reserved
8	A3		Technology Ability Field [3]
		0	The Ethernet PHY does not support the 100Base-TX full-duplex signaling protocol.
		1	The Ethernet PHYr supports the 100Base-TX full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MR0 register.
7	A2		Technology Ability Field [2]
		0	The Ethernet PHY does not support the 100Base-TX half-duplex signaling protocol.
		1	The Ethernet PHY supports the 100Base-TX half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MR0 register.
6	A1		Technology Ability Field [1]
		0	The Ethernet PHY does not support the 10BASE-T full-duplex signaling protocol.
		1	The Ethernet PHY supports the 10BASE-T full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MR0 register.
5	A0		Technology Ability Field [0]
		0	The Ethernet PHY does not support the 10BASE-T half-duplex signaling protocol.
		1	The Ethernet PHY supports the 10BASE-T half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MR0 register.
4-0	S		Selector Field
		1h	This field encodes 32 possible messages for communicating between Ethernet PHYs.

Copyright © 2012–2019, Texas Instruments Incorporated

19.7.6 Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5) Register, address 0x05

The Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5) register provides the advertised abilities of the link partner's Ethernet PHY that are received and stored during auto-negotiation.

Figure 19-26. Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5) Register

15	14	13	12	9	8
NP	ACK	RF	A		
R-0	R-0	R-0	R-0		

7	5	4	0
A		S	
R-0		R-1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-25. Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5) Register Field Descriptions

Bit	Field	Value	Description
15	NP		Next Page
		0	The link partner's Ethernet PHY is not capable of Next Page exchanges.
		1	The link partner's Ethernet PHY is capable of Next Page exchanges to provide more detailed information on the PHY's capabilities.
14	ACK		Acknowledge
		0	The Ethernet PHY has not received the link partner's advertised abilities during auto-negotiation.
		1	The Ethernet PHY has successfully received the link partner's advertised abilities during auto-negotiation.
13	RF		Remote Fault
		0	The link partner is not indicating that a Remote Fault condition has been encountered.
		1	The link partner is indicating that a Remote Fault condition has been encountered.
12-5	A		ReservedTechnology Ability Field This field encodes individual technologies that are supported by the Ethernet PHY. See the MR4 register for definitions. Refer to the IEEE 802.3 standard for definitions.
4-0	S		Selector Field
			This field encodes possible messages for communicating between Ethernet PHYs.
		00	Reserved
		01	IEEE Std 802.3
		02	IEEE Std 802.9 ISLAN-16T
		03	IEEE Std 802.5
		04	IEEE Std 1394
		05-1F	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

19.7.7 Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6) Register, address 0x06

The Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6) register enables software to determine the auto-negotiation and next page capabilities of the Ethernet PHY and the link partner after auto-negotiation.

Figure 19-27. Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6) Register

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved			PDF	LPNPA	Reserved	PRX	LPANEGA
R-0			R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-26. Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6) Register Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved
4	PDF		
		0	Only one technology was detected at link up.
		1	More than one technology was detected at link up. This bit is automatically cleared when read.
3	LPNPA		Link Partner is Next Page Able
		0	The link partner is not enabled to support next page.
		1	The link partner is enabled to support next page.
2	Reserved		Reserved
1	PRX		New Page Received
		0	A new page has not been received.
		1	A new page has been received from the link partner and stored. This bit is automatically cleared when read.
0	LPANEGA		Link Partner is Auto-Negotiation Able
		0	The link partner is not enabled to support auto-negotiation
		1	The link partner is enabled to support auto-negotiation.

Copyright © 2012–2019, Texas Instruments Incorporated

M3 Synchronous Serial Interface (SSI)

This chapter describes synchronous serial interface (SSI) modules. Refer to the device datasheet for the number of instances present on your device.

Topic **Page**

20.1 Introduction **1410**
20.2 Features **1410**
20.3 Functional Description **1411**
20.4 Initialization and Configuration **1418**
20.5 SSI Registers **1419**

Copyright © 2012–2019, Texas Instruments Incorporated

20.1 Introduction

Each SSI is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, or Texas Instruments Synchronous Serial Interfaces.

20.2 Features

The SSI module includes the following features:

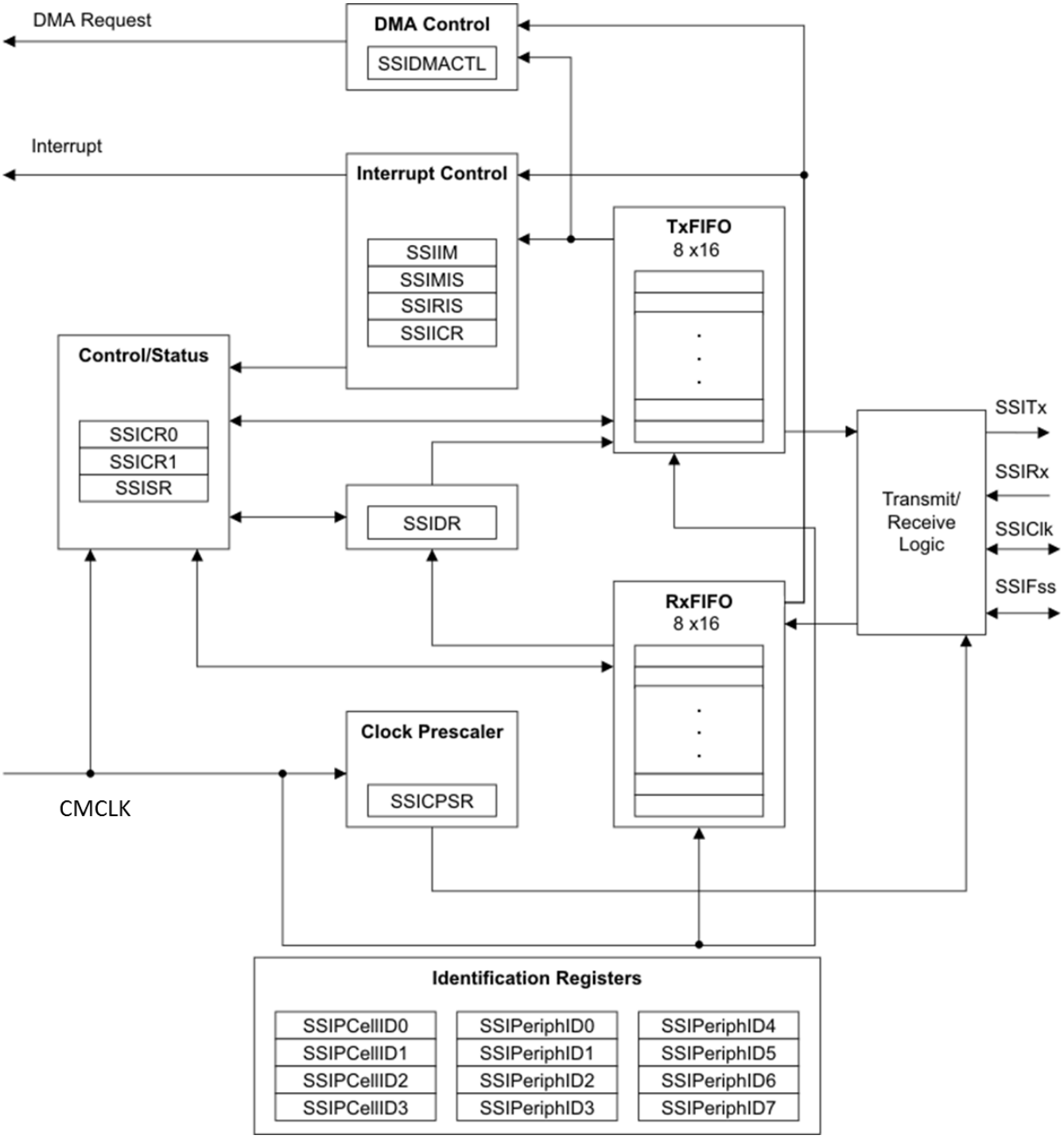
- Programmable interface operation for Freescale SPI, or Texas Instruments Synchronous Serial Interfaces.
- Master or slave operation
- Programmable clock bit rate and prescaler
- Separate transmit and receive FIFOs, each 16 bits wide and 8 locations deep
- Programmable data frame size from 4 to 16 bits
- 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 four entries
  - Transmit single request asserted when there is space in the FIFO; burst request asserted when FIFO contains four or more entries are available to be written in the FIFO
  - Maskable µDMA interrupts for receive and transmit complete

20.2.1 SSI Block Diagram

The SSI module block diagram is shown in the figure below.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 20-1. SSI Block Diagram



20.3 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.

Copyright © 2012–2019, Texas Instruments Incorporated

20.3.1 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 (CMCLK). 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. 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.

The frequency of the output clock SSIClk is defined by:

SSIClk = CMCLK / (CPSDVSR * (1 + SCR))

NOTE: For master mode, the system clock must be at least two times faster than the SSIClk, with the restriction that SSIClk cannot be faster than (CMCLK/2). For slave mode, the system clock must be at least 12 times faster than the SSIClk. In slave mode, maximum frequency of operation is (CMCLK/12).

See the *Electrical Characteristics* chapter in the device datasheet to view SSI timing parameters.

20.3.2 FIFO Operation

20.3.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the SSI Data (SSIDR) register, and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master, respectively, through the SSITx pin.

In slave mode, the SSI transmits data each time the master initiates a transaction. If the transmit FIFO is empty and the master initiates, the slave transmits the eighth most recent value in the transmit FIFO. If less than eight values have been written to the transmit FIFO since the SSI module clock was enabled using the SSI bit in the CMPCLKCR0 register, then 0 is transmitted. Take care 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.

20.3.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the SSIDR register.

When configured as a master or slave, serial data received through the SSIRx pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

When receiving data in master mode, a dummy write to the SSIDR register must occur before any read occurs.

20.3.3 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

Copyright © 2012–2019, Texas Instruments Incorporated

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. 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.

The receive FIFO has a time-out period that is 32 periods at the rate of SSIClk (whether or not SSIClk is currently active) and is started when the RX FIFO goes from EMPTY to not-EMPTY. If the RX FIFO is emptied before 32 clocks have passed, the time-out period is reset. As a result, the ISR should clear the Receive FIFO Time-out Interrupt just after reading out the RX FIFO by writing a 1 to the RTIC bit in the SSI Interrupt Clear (SSIICR) register. The interrupt should not be cleared so late that the ISR returns before the interrupt is actually cleared, or the ISR may be re-activated unnecessarily.

The End-of-Transmission (EOT) interrupt indicates that the data has been transmitted completely. This interrupt can be used to indicate when it is safe to turn off the SSI module clock or enter sleep mode. In addition, because transmitted data and received data complete at exactly the same time, the interrupt can also indicate that read data is ready immediately, without waiting for the receive FIFO time-out period to complete.

20.3.4 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. Two basic frame types can be selected:

- Texas Instruments synchronous serial
- Freescale SPI

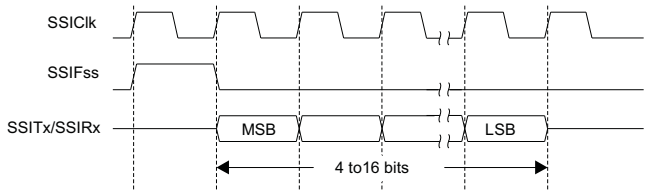
For all three formats, the serial clock (SSIClk) is held inactive while the SSI is idle, and SSIClk transitions at the programmed frequency only during active transmission or reception of data. The idle state of SSIClk is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI frame format, the serial frame (SSIFss) pin is active low, and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the SSIFss pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of SSIClk and latch data from the other device on the falling edge.

The figure below shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 20-2. TI Synchronous Serial Frame Format (Single Transfer)



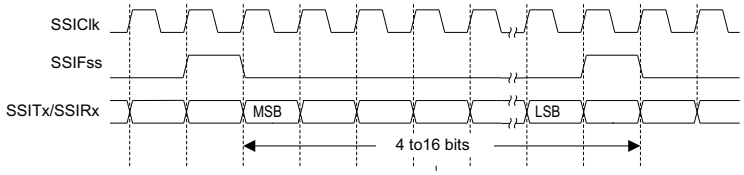
In this mode, SSIClk and SSIFss are forced low, and the transmit data line SSITx is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SSIFss is pulsed high for one SSIClk period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSIClk, the MSB of the 4- to 16-bit data frame is shifted out on the SSITx pin. Likewise, the MSB of the received data is shifted onto the SSIRx pin by the off-chip serial slave device.

Copyright © 2012–2019, Texas Instruments Incorporated

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on each falling edge of SSIClk. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSIClk after the LSB has been latched.

The figure below shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 20-3. TI Synchronous Serial Frame Format (Continuous Transfer)



20.3.4.1 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the SSIFss signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the SSIClk signal are programmable through the SPO and SPH bits in the SSICR0 control register.

20.3.4.1.1 SPO Clock Polarity Bit

When the SPO clock polarity control bit is clear, it produces a steady state low value on the SSIClk pin. If the SPO bit is set, a steady state high value is placed on the SSIClk pin when data is not being transferred.

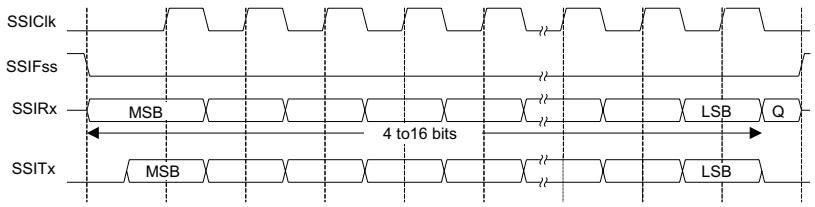
20.3.4.1.2 SPH Phase Control Bit

The SPH phase control bit selects the clock edge that captures data and allows it to change state. The state of this bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is clear, data is captured on the first clock edge transition. If the SPH bit is set, data is captured on the second clock edge transition.

20.3.4.2 Freescale SPI Frame Format with SPO=0 and SPH=0

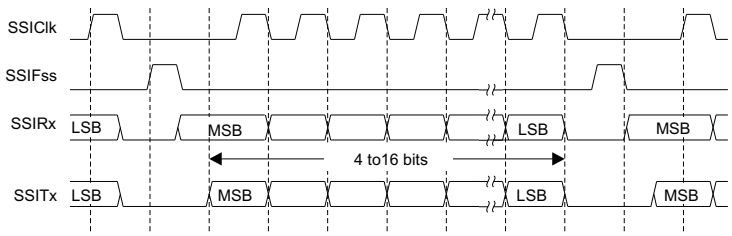
Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 20-4 and Figure 20-5

Figure 20-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0



Note: Q is undefined.

Figure 20-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0



Copyright © 2012–2019, Texas Instruments Incorporated

In this configuration, during idle periods:

- SSIClk is forced low
- SSIFss is forced high
- The transmit data line SSITx is arbitrarily forced low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven low, causing slave data to be enabled onto the SSIRx input line of the master. The master SSITx output pad is enabled.

One half SSIClk period later, valid master data is transferred to the SSITx pin. Once both the master and slave data have been set, the SSIClk master clock pin goes high after one additional half SSIClk period. The data is now captured on the rising and propagated on the falling edges of the SSIClk signal.

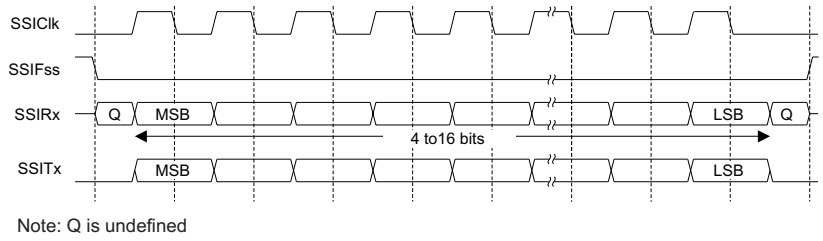
In the case of a single word transmission, after all bits of the data word have been transferred, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed high between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is clear. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

20.3.4.3 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 the figure below, in which covers both single and continuous transfers.

Figure 20-6. Freescale SPI Frame Format with SPO =0 and SPH=1



Note: Q is undefined

In this configuration, during idle periods:

- SSIClk is forced low
- SSIFss is forced high
- The transmit data line SSITx is arbitrarily forced low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven low. The master SSITx output is then enabled. After an additional one-half SSIClk period, both master and slave valid data are enabled onto their respective transmission lines. At the same time, the SSIClk is enabled with a rising edge transition. Data is then captured on the falling edges and propagated on the rising edges of the SSIClk signal.

In the case of a single word transfer, after all bits have been transferred, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

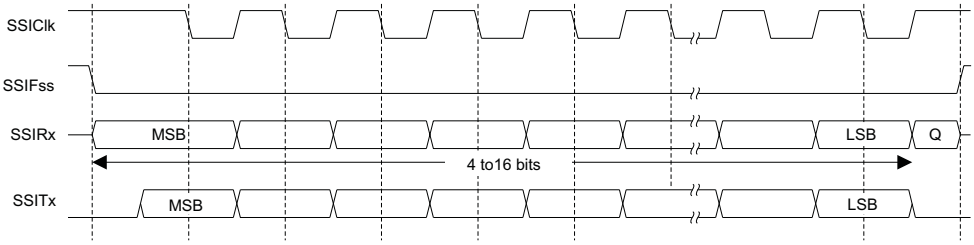
For continuous back-to-back transfers, the SSIFss pin is held low between successive data words, and termination is the same as that of the single word transfer.

Copyright © 2012–2019, Texas Instruments Incorporated

20.3.4.4 Freescale SPI Frame Format with SPO=1 and SPH=0

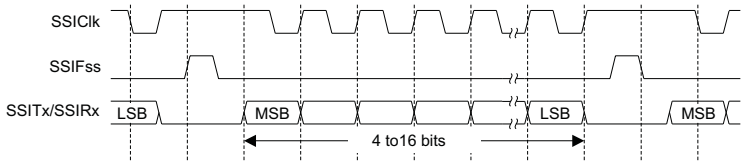
Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 20-7 and Figure 20-8

Figure 20-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0



Note: Q is undefined.

Figure 20-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0



In this configuration, during idle periods:

- SSIClk is forced high
- SSIFss is forced high
- The transmit data line SSITx is arbitrarily forced low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven low, causing slave data to be immediately transferred onto the SSIRx line of the master. The master SSITx output pad is enabled.

One-half period later, valid master data is transferred to the SSITx line. Once both the master and slave data have been set, the SSIClk master clock pin becomes low after one additional half SSIClk period, meaning that data is captured on the falling edges and propagated on the rising edges of the SSIClk signal.

In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

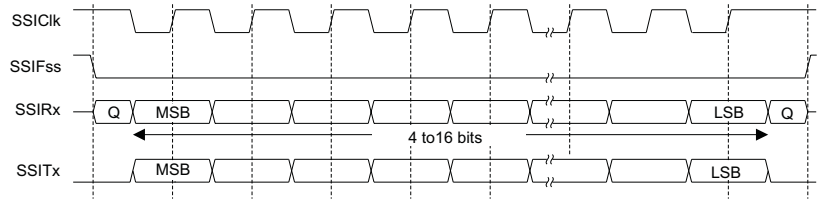
However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed high between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is clear. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

20.3.4.5 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 20-9 which covers both single and continuous transfers.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 20-9. Freescale SPI Frame Format with SPO =1 and SPH =1



In this configuration, during idle periods:

- SSIClk is forced hgh
- SSIFss is forced high
- The transmit data line SSITx is arbitrarily forced low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven low. The master SSITx output pad is enabled. After an additional one-half SSIClk period, both master and slave data are enabled onto their respective transmission lines. At the same time, SSIClk is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSIClk signal.

After all bits have been transferred, in the case of a single word transmission, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

For continuous back-to-back transmissions, the SSIFss pin remains in its active low state until the final bit of the last word has been captured and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSIFss pin is held low between successive data words and termination is the same as that of the single word transfer.

20.3.5 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 four 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 four or more empty slots. The single and burst µDMA transfer requests are handled automatically by the µDMA controller depending how the µDMA channel is configured. To enable µDMA operation for the receive channel, the RXDMAE bit of the DMA Control (SSIDMACTL) register should be set. To enable µDMA operation for the transmit channel, the TXDMAE bit of SSIDMACTL should be set. If µDMA is enabled, then the µDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the SSI interrupt vector. Therefore, if interrupts are used for SSI operation and µDMA is enabled, the SSI interrupt handler must be designed to handle the µDMA completion interrupt.

If the µDMA is enabled and has completed a data transfer from the Tx FIFO, the DMATXRIS bit is set in the SSIRIS register and cannot be cleared by setting the DMATXIC bit in the SSI Interrupt Clear (SSIICR) register. In the DMA Completion Interrupt Service Routine, software must disable the µDMA transmit enable to the SSI by clearing the TXDMAE bit in the SSI DMA Control (SSIDMACTL) register and then setting the DMATXIC bit in the SSIICR register. This clears the DMA completion interrupt. When the µDMA is needed to transmit more data, the TXDMAE bit must be set (enabled) again. If a data transfer by the µDMA from the Rx FIFO completes, the DMARXRIS bit is set. The EOT bit in the SSIRIS register is also provided to indicate when the Tx FIFO is empty and the last bit has been transmitted out of the serializer.

Copyright © 2012–2019, Texas Instruments Incorporated

See the *Micro Direct Memory Access (µDMA)* chapter for more details about programming the µDMA controller.

20.4 Initialization and Configuration

To enable and initialize the SSI, the following steps are necessary:

1. Enable the SSI module by setting the SSI bit in the CMPCLKCR0 register.
2. Configure the pinmux.

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:
   - For master operations, set the SSICR1 register to 0x0000.0000.
   - For slave mode (output enabled), set the SSICR1 register to 0x0000.0004.
   - For slave mode (output disabled), set the SSICR1 register to 0x0000.000C.
3. Configure the clock prescale divisor by writing the SSICPSR register.
4. Write the SSICR0 register with the following configuration:
   - Serial clock rate (SCR)
   - Desired clock phase/polarity, if using Freescale SPI mode (SPH and SPO)
   - Protocol mode: Freescale SPI, TI SSF
   - Data size (DSS)
5. Optionally, configure the µDMA channel (see the *Micro Direct Memory Access (µDMA)* chapter) and enable the DMA option(s) in the SSIDMACTL register.
6. Enable the SSI by setting the SSE bit in the SSICR1 register.

As an example, assume the SSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (SPO=1, SPH=1)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock (CMCLK) is 20 MHz, the bit rate calculation would be:

SSIClk = (system clock) / (CPSDVSR * (1 + SCR)) 1x106 = 20x106 / (CPSDVSR * (1 + SCR))

In this case, if CPSDVSR=0x2, SCR must be 0x9.

The configuration sequence would be as follows:

1. Ensure that the SSE bit in the SSICR1 register is clear.
2. Write the SSICR1 register with a value of 0x0000.0000.
3. Write the SSICPSR register with a value of 0x0000.0002.
4. Write the SSICR0 register with a value of 0x0000.09C7.
5. The SSI is then enabled by setting the SSE bit in the SSICR1 register.

Copyright © 2012–2019, Texas Instruments Incorporated

20.5 SSI Registers

This section describes the Synchronous Serial Interface registers.

20.5.1 SSI Base Addresses

Table 20-1. SSI Base Address Table (CM)

DriverLib Name	Base Address	uDMA Access	Ethernet DMA Access
SSI0_BASE	0x4000_8000	YES	-

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2 SSI_REGS Registers

Table 20-2 lists the SSI_REGS registers. All register offset addresses not listed in Table 20-2 should be considered as reserved locations and the register contents should not be modified.

Table 20-2. SSI_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	SSICR0	SSI Control 0		Go
4h	SSICR1	SSI Control 1		Go
8h	SSIDR	SSI Data		Go
Ch	SSISR	SSI Status		Go
10h	SSICPSR	SSI Clock Prescale		Go
14h	SSIIM	SSI Interrupt Mask		Go
18h	SSIRIS	SSI Raw Interrupt Status		Go
1Ch	SSIMIS	SSI Masked Interrupt Status		Go
20h	SSIICR	SSI Interrupt Clear		Go
24h	SSIDMACTL	SSI DMA Control		Go
FB0h	SSIPV	SSI Peripheral Version		Go
FC0h	SSIPP	SSI Peripheral Properties		Go
FC4h	SSIPC	SSI Peripheral Configuration		Go
FD0h	SSIPeriphID4	SSI Peripheral Identification 4		Go
FD4h	SSIPeriphID5	SSI Peripheral Identification 5		Go
FD8h	SSIPeriphID6	SSI Peripheral Identification 6		Go
FDCh	SSIPeriphID7	SSI Peripheral Identification 7		Go
FE0h	SSIPeriphID0	SSI Peripheral Identification 0		Go
FE4h	SSIPeriphID1	SSI Peripheral Identification 1		Go
FE8h	SSIPeriphID2	SSI Peripheral Identification 2		Go
FECh	SSIPeriphID3	SSI Peripheral Identification 3		Go
FF0h	SSIPCellID0	SSI PrimeCell Identification 0		Go
FF4h	SSIPCellID1	SSI PrimeCell Identification 1		Go
FF8h	SSIPCellID2	SSI PrimeCell Identification 2		Go
FFCh	SSIPCellID3	SSI PrimeCell Identification 3		Go

Complex bit access types are encoded to fit into small table cells. Table 20-3 shows the codes that are used for access types in this section.

Table 20-3. SSI_REGS Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
W1S	W 1S	Write 1 to set
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		

Copyright © 2012–2019, Texas Instruments Incorporated

Table 20-3. SSI_REGS Access Type Codes (continued)

Access Type	Code	Description
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.1 SSICR0 Register (Offset = 0h) [reset = 0h]

SSICR0 is shown in Figure 20-10 and described in Table 20-4.

Return to the Summary Table.

SSI Control 0

Figure 20-10. SSICR0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCR								SPH	SPO	FRF		DSS			
R/W-0h								R/W-0h	R/W-0h	R/W-0h		R/W-0h			

Table 20-4. SSICR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	SCR	R/W	0h	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. Reset type: PER.RESET
7	SPH	R/W	0h	SSI Serial clock PHase This bit is only applicable to the Freescale SPI Format.The 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. SPH Value Description 0 Data is captured on the first clock edge transition. 1 Data is captured on the second clock edge transition. Reset type: PER.RESET
6	SPO	R/W	0h	SSI Serial clock POlarity Value Description 0 A steady state Low value is placed on the pin SSInClk when data is not being transferred. 1 A steady state High value is placed on the pin SSInClk when data is not being transferred. Reset type: PER.RESET
5-4	FRF	R/W	0h	SSI FRame Format Select Value Frame Format 0x0 Freescale SPI Frame Format 0x1 Synchronous Serial Frame Format Texas Instruments 0x2 Reserved 0x3 Reserved Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

Table 20-4. SSICR0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	DSS	R/W	0h	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 Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.2 SSICR1 Register (Offset = 4h) [reset = 0h]

SSICR1 is shown in Figure 20-11 and described in Table 20-5.

Return to the Summary Table.

SSI Control 1

Figure 20-11. SSICR1 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED					FSSHLDFRM	HSCLKEN	DIR
R-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
			EOT		MS	SSE	LBM
			R/W-0h		R/W-0h	R/W-0h	R/W-0h

Table 20-5. SSICR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	Reserved
10	FSSHLDFRM	R/W	0h	FSS Hold Frame Value Description 0 Pulse SSInFss at every byte (the bit DSS in the SSICR0 register must be set to 0x7 (data size 8 bits) in thisconfiguration) 1 Hold SSInFss for the whole frame Reset type: PER.RESET
9	HSCLKEN	R/W	0h	High Speed Clock Enable High speed clock enable is available only when operating as a master. Value Description 0 Use Input Clock 1 Use High Speed Clock Note: For proper functionality of high speed mode, the HSCLKEN bit in the SSICR1 register should be set before any SSI data transfer or after applying a reset to the QSSI module. In addition, the SSE bit must be set to 0x1 before the HSCLKEN bit is set. Reset type: PER.RESET
8	DIR	R/W	0h	SSI Direction of Operation Value Description 0 TX (Transmit Mode) write direction 1 RX (Receive Mode) read direction Reset type: PER.RESET
4	EOT	R/W	0h	End of Transmission This bit is only valid for Master mode devices and operations (=0x0).MS 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.When using uDMA, the DMATX bit cannot be set to 1 in any mode. If used with uDMA, it prevents RISEOT from asserting. If the bit is kept at 0 during operation, an interrupt is still generated when the TX FIFO is half or less full with or without using the uDMA.EOT (Legacy) Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

Table 20-5. SSICR1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	MS	R/W	0h	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. Reset type: PER.RESET
1	SSE	R/W	0h	SSI Synchronous Serial Port Enable Value Description 0 SSI operation is disabled. 1 SSI operation is enabled. This bit must be cleared before any control registers are reprogrammed. The bit HSCLKEN in the SSICR1 register should be set only after applying reset to the QSSI module and enabling the QSSI by setting the SSE bit, and before any SSI data transfer. All other bits in the SSICR1 register and all bits in SSICR0 register can only be programmed when the SSE is clear. Reset type: PER.RESET
0	LBM	R/W	0h	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. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.3 SSIDR Register (Offset = 8h) [reset = 0h]

SSIDR is shown in Figure 20-12 and described in Table 20-6.

Return to the Summary Table.

SSI Data

Figure 20-12. SSIDR Register

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

Table 20-6. SSIDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	DATA	R/W	0h	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. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.4 SSISR Register (Offset = Ch) [reset = 3h]

SSISR is shown in Figure 20-13 and described in Table 20-7.

Return to the Summary Table.

SSI Status

Figure 20-13. SSISR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											BSY	RFF	RNE	TNF	TFE
R-0h											R-0h	R-0h	R-0h	R-1h	R-1h

Table 20-7. SSISR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	BSY	R	0h	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. Reset type: PER.RESET
3	RFF	R	0h	SSI Receive FIFO Full Value Description 0 The receive FIFO is not full. 1 The receive FIFO is full. Reset type: PER.RESET
2	RNE	R	0h	SSI Receive FIFO Not Empty Value Description 0 The receive FIFO is empty. 1 The receive FIFO is not empty. Reset type: PER.RESET
1	TNF	R	1h	SSI Transmit FIFO Not Full Value Description 0 The transmit FIFO is full. 1 The transmit FIFO is not full. Reset type: PER.RESET
0	TFE	R	1h	SSI Transmit FIFO Empty Value Description 0 The transmit FIFO is not empty. 1 The transmit FIFO is empty. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.5 SSICPSR Register (Offset = 10h) [reset = 0h]

SSICPSR is shown in Figure 20-14 and described in Table 20-8.

Return to the Summary Table.

SSI Clock Prescale

Figure 20-14. SSICPSR Register

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

Table 20-8. SSICPSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	CPSDVSR	R/W	0h	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. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.6 SSIIM Register (Offset = 14h) [reset = 0h]

SSIIM is shown in Figure 20-15 and described in Table 20-9.

Return to the Summary Table.

SSI Interrupt Mask

Figure 20-15. SSIIM Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	EOTIM	DMATXIM	DMARXIM	TXIM	RXIM	RTIM	RORIM
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 20-9. SSIIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6	EOTIM	R/W	0h	End of Transmit Interrupt Mask Value Description 0 The end of transmit interrupt is masked. 1 The end of transmit interrupt is not masked. Reset type: PER.RESET
5	DMATXIM	R/W	0h	SSI Transmit DMA Interrupt Mask Value Description 0 The transmit DMA interrupt is masked. 1 The transmit DMA interrupt is not masked. Reset type: PER.RESET
4	DMARXIM	R/W	0h	SSI Receive DMA Interrupt Mask Value Description 0 The receive DMA interrupt is masked. 1 The receive DMA interrupt is not masked. Reset type: PER.RESET
3	TXIM	R/W	0h	SSI Transmit FIFO Interrupt Mask Value Description 0 The transmit FIFO interrupt is masked. 1 The transmit FIFO interrupt is not masked. Reset type: PER.RESET
2	RXIM	R/W	0h	SSI Receive FIFO Interrupt Mask Value Description 0 The receive FIFO interrupt is masked. 1 The receive FIFO interrupt is not masked. Reset type: PER.RESET
1	RTIM	R/W	0h	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. Reset type: PER.RESET
0	RORIM	R/W	0h	SSI Receive Overrun Interrupt Mask Value Description 0 The receive FIFO overrun interrupt is masked. 1 The receive FIFO overrun interrupt is not masked. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.7 SSIRIS Register (Offset = 18h) [reset = 8h]

SSIRIS is shown in Figure 20-16 and described in Table 20-10.

Return to the Summary Table.

SSI Raw Interrupt Status

Figure 20-16. SSIRIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	EOTRIS	DMATXRIS	DMARXRIS	TXRIS	RXRIS	RTRIS	RORRIS
R-0h	R-0h	R-0h	R-0h	R-1h	R-0h	R-0h	R-0h

Table 20-10. SSIRIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6	EOTRIS	R	0h	End of Transmit Raw Interrupt Status Value Description 0 No interrupt. 1 The transmit FIFO is empty, and the last bit has been transmitted out of the serializer.This bit is cleared when a 1 is written to the EOTIC bit in the SSI Interrupt Clear (SSIICR) register. Reset type: PER.RESET
5	DMATXRIS	R	0h	SSI Transmit DMA Raw Interrupt Status Value Description 0 No interrupt. 1 The transmit DMA has completed.This bit is cleared when a 1 is written to the DMATXIC bit in the SSI Interrupt Clear (SSIICR) register. Reset type: PER.RESET
4	DMARXRIS	R	0h	SSI Receive DMA Raw Interrupt Status Value Description 0 No interrupt. 1 The receive DMA has completed.This bit is cleared when a 1 is written to the DMARXIC bit in the SSI Interrupt Clear (SSIICR) register. Reset type: PER.RESET
3	TXRIS	R	1h	SSI Transmit FIFO Raw Interrupt Status Value Description 0 No interrupt. 1 The transmit FIFO is half empty or less. If the EOT bit in the SSICR1 register is clear, 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) Reset type: PER.RESET
2	RXRIS	R	0h	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. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

Table 20-10. SSIRIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	RTRIS	R	0h	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. Reset type: PER.RESET
0	RORRIS	R	0h	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. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.8 SSIMIS Register (Offset = 1Ch) [reset = 0h]

SSIMIS is shown in Figure 20-17 and described in Table 20-11.

Return to the Summary Table.

SSI Masked Interrupt Status

Figure 20-17. SSIMIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	EOTMIS	DMATXMIS	DMARXMIS	TXMIS	RXMIS	RTMIS	RORMIS
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

Table 20-11. SSIMIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6	EOTMIS	R	0h	End of Transmit Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the transmission of the last data bit.This bit is cleared when a 1 is written to the EOTIC bit in the SSI Interrupt Clear (SSIICR) register. Reset type: PER.RESET
5	DMATXMIS	R	0h	SSI Transmit DMA Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due tothe completion of the transmit DMA.This bit is cleared when a 1 is written to the DMATXIC bit in the SSI Interrupt Clear (SSIICR) register. Reset type: PER.RESET
4	DMARXMIS	R	0h	SSI Receive DMA Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the completion of the receive DMA.This bit is cleared when a 1 is written to the DMARXIC bit in the SSI Interrupt Clear (SSIICR) register. Reset type: PER.RESET
3	TXMIS	R	0h	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) Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

Table 20-11. SSIMIS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	RXMIS	R	0h	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. Reset type: PER.RESET
1	RTMIS	R	0h	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. Reset type: PER.RESET
0	RORMIS	R	0h	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. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.9 SSIICR Register (Offset = 20h) [reset = 0h]

SSIICR is shown in Figure 20-18 and described in Table 20-12.

Return to the Summary Table.

SSI Interrupt Clear

Figure 20-18. SSIICR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	EOTIC	DMATXIC	DMARXIC	RESERVED		RTIC	RORIC
R-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0h		R-0/W1S-0h	R-0/W1S-0h

Table 20-12. SSIICR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R	0h	Reserved
6	EOTIC	R-0/W1S	0h	End of Transmit Interrupt Clear Writing a 1 to this bit clears the EOTRIS bit in the SSIRIS register and the EOTMIS bit in the SSIMIS register. Reset type: PER.RESET
5	DMATXIC	R-0/W1S	0h	SSI Transmit DMA Interrupt Clear Writing a 1 to this bit clears the DMATXRIS bit in the SSIRIS register and the DMATXMIS bit in the SSIMIS register. Reset type: PER.RESET
4	DMARXIC	R-0/W1S	0h	SSI Receive DMA Interrupt Clear Writing a 1 to this bit clears the DMARXRIS bit in the SSIRIS register and the DMARXMIS bit in the SSIMIS register. Reset type: PER.RESET
3-2	RESERVED	R	0h	Reserved
1	RTIC	R-0/W1S	0h	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. Reset type: PER.RESET
0	RORIC	R-0/W1S	0h	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. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.10 SSIDMACTL Register (Offset = 24h) [reset = 0h]

SSIDMACTL is shown in Figure 20-19 and described in Table 20-13.

Return to the Summary Table.

SSI DMA Control

Figure 20-19. SSIDMACTL Register

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

Table 20-13. SSIDMACTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	TXDMAE	R/W	0h	Transmit DMA Enable Value Description 0 uDMA for the transmit FIFO is disabled. 1 uDMA for the transmit FIFO is enabled. Reset type: PER.RESET
0	RXDMAE	R/W	0h	Receive DMA Enable Value Description 0 uDMA for the receive FIFO is disabled. 1 uDMA for the receive FIFO is enabled. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.11 SSIPV Register (Offset = FB0h) [reset = 400h]

SSIPV is shown in Figure 20-20 and described in Table 20-14.

Return to the Summary Table.

SSI Peripheral Version

Figure 20-20. SSIPV Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																MAJOR								MINOR							
R-0h																R-4h								R-0h							

Table 20-14. SSIPV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	MAJOR	R	4h	Major Revision This field specifies the major revision number of the module. Corresponds to the IP version used on Snowflake. Reset type: PER.RESET
7-0	MINOR	R	0h	Minor Revision This field specifies the minor revision number of the module. Corresponds to the IP version used on Snowflake. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.12 SSIPP Register (Offset = FC0h) [reset = 9h]

SSIPP is shown in Figure 20-21 and described in Table 20-15.

Return to the Summary Table.

SSI Peripheral Properties

Figure 20-21. SSIPP Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				FSSHLDFRM	MODE		HSCLK
R-0h				R-1h	R-0h		R-1h

Table 20-15. SSIPP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
3	FSSHLDFRM	R	1h	SSInFss Hold Frame Capability Value Description 0 Hold Frame capability disabled.SSInFss 1 Hold Frame capability enabled.SSinFss Reset type: PER.RESET
2-1	MODE	R	0h	Mode of Operation Indicates what SSI functionality is supported. Value Description 0x0 Legacy SSI mode Others reserved Reset type: PER.RESET
0	HSCLK	R	1h	High Speed Capability Value Description 0 High Speed clock capability disabled. 1 High speed clock capability enabled. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.13 SSIPC Register (Offset = FC4h) [reset = 0h]

SSIPC is shown in Figure 20-22 and described in Table 20-16.

Return to the Summary Table.

SSI Peripheral Configuration

Figure 20-22. SSIPC Register

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

Table 20-16. SSIPC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.14 SSIPeriphID4 Register (Offset = FD0h) [reset = 0h]

SSIPeriphID4 is shown in Figure 20-23 and described in Table 20-17.

Return to the Summary Table.

SSI Peripheral Identification 4

Figure 20-23. SSIPeriphID4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID4							
R-0h																								R-0h							

Table 20-17. SSIPeriphID4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PID4	R	0h	SSI Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.15 SSIPeriphID5 Register (Offset = FD4h) [reset = 0h]

SSIPeriphID5 is shown in Figure 20-24 and described in Table 20-18.

Return to the Summary Table.

SSI Peripheral Identification 5

Figure 20-24. SSIPeriphID5 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID5							
R-0h																								R-0h							

Table 20-18. SSIPeriphID5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PID5	R	0h	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.16 SSIPeriphID6 Register (Offset = FD8h) [reset = 0h]

SSIPeriphID6 is shown in Figure 20-25 and described in Table 20-19.

Return to the Summary Table.

SSI Peripheral Identification 6

Figure 20-25. SSIPeriphID6 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID6							
R-0h																								R-0h							

Table 20-19. SSIPeriphID6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PID6	R	0h	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.17 SSIPeriphID7 Register (Offset = FDCh) [reset = 0h]

SSIPeriphID7 is shown in Figure 20-26 and described in Table 20-20.

Return to the Summary Table.

SSI Peripheral Identification 7

Figure 20-26. SSIPeriphID7 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID7							
R-0h																								R-0h							

Table 20-20. SSIPeriphID7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PID7	R	0h	SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.18 SSIPeriphID0 Register (Offset = FE0h) [reset = 22h]

SSIPeriphID0 is shown in Figure 20-27 and described in Table 20-21.

Return to the Summary Table.

SSI Peripheral Identification 0

Figure 20-27. SSIPeriphID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID0							
R-0h																								R-22h							

Table 20-21. SSIPeriphID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PID0	R	22h	SSI Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.19 SSIPeriphID1 Register (Offset = FE4h) [reset = 0h]

SSIPeriphID1 is shown in Figure 20-28 and described in Table 20-22.

Return to the Summary Table.

SSI Peripheral Identification 1

Figure 20-28. SSIPeriphID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID1							
R-0h																								R-0h							

Table 20-22. SSIPeriphID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PID1	R	0h	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.20 SSIPeriphID2 Register (Offset = FE8h) [reset = 18h]

SSIPeriphID2 is shown in Figure 20-29 and described in Table 20-23.

Return to the Summary Table.

SSI Peripheral Identification 2

Figure 20-29. SSIPeriphID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID2							
R-0h																								R-18h							

Table 20-23. SSIPeriphID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PID2	R	18h	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.21 SSIPeriphID3 Register (Offset = FECh) [reset = 1h]

SSIPeriphID3 is shown in Figure 20-30 and described in Table 20-24.

Return to the Summary Table.

SSI Peripheral Identification 3

Figure 20-30. SSIPeriphID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PID3							
R-0h																								R-1h							

Table 20-24. SSIPeriphID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	PID3	R	1h	SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.22 SSIPCellID0 Register (Offset = FF0h) [reset = Dh]

SSIPCellID0 is shown in Figure 20-31 and described in Table 20-25.

Return to the Summary Table.

SSI PrimeCell Identification 0

Figure 20-31. SSIPCellID0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID0							
R-0h																								R-Dh							

Table 20-25. SSIPCellID0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	CID0	R	Dh	SSI PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.23 SSIPCellID1 Register (Offset = FF4h) [reset = F0h]

SSIPCellID1 is shown in Figure 20-32 and described in Table 20-26.

Return to the Summary Table.

SSI PrimeCell Identification 1

Figure 20-32. SSIPCellID1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID1							
R-0h																								R-F0h							

Table 20-26. SSIPCellID1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	CID1	R	F0h	SSI PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.24 SSIPCellID2 Register (Offset = FF8h) [reset = 5h]

SSIPCellID2 is shown in Figure 20-33 and described in Table 20-27.

Return to the Summary Table.

SSI PrimeCell Identification 2

Figure 20-33. SSIPCellID2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID2							
R-0h																								R-5h							

Table 20-27. SSIPCellID2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	CID2	R	5h	SSI PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

20.5.2.25 SSIPCellID3 Register (Offset = FFCh) [reset = B1h]

SSIPCellID3 is shown in Figure 20-34 and described in Table 20-28.

Return to the Summary Table.

SSI PrimeCell Identification 3

Figure 20-34. SSIPCellID3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CID3							
R-0h																								R-B1h							

Table 20-28. SSIPCellID3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	CID3	R	B1h	SSI PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system. Reset type: PER.RESET

Copyright © 2012–2019, Texas Instruments Incorporated

M3 Universal Asynchronous Receivers/Transmitters (UARTs)

This chapter discusses the features and functions of the Universal Asynchronous Receiver/Transmitter (UART) module.

Topic **Page**

21.1 Introduction 1452
21.2 Block Diagram 1452
21.3 Functional Description 1453
21.4 M3 UART4 to C28 SCI-A Internal Loopback 1458
21.5 Initialization and Configuration 1459
21.6 Register Map 1460
21.7 Register Descriptions 1461

Copyright © 2012–2019, Texas Instruments Incorporated

21.1 Introduction

The Universal Asynchronous Receiver/Transmitter (UART) includes the following features:

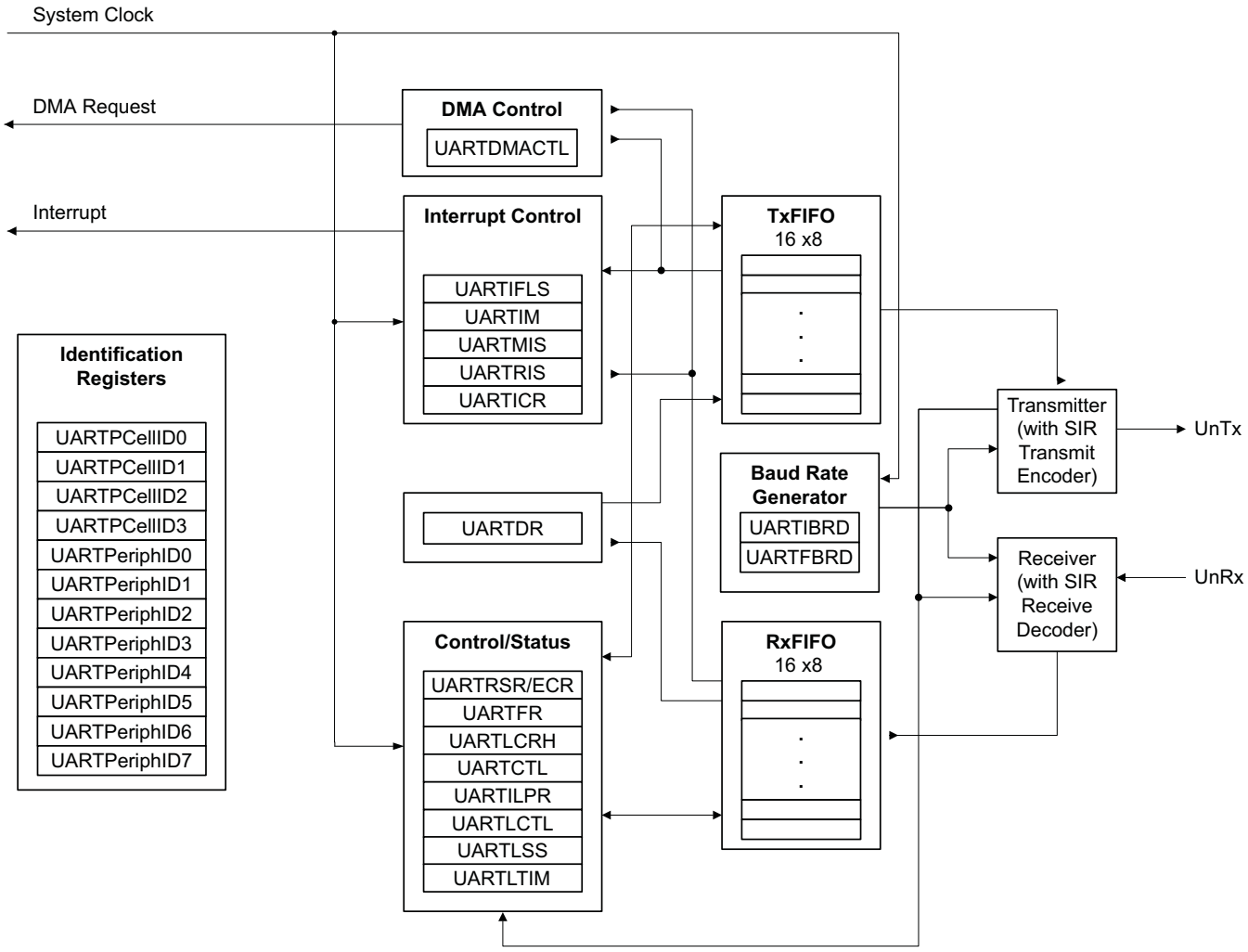
- Programmable baud-rate generator allowing speeds up to M3 System clock / 16 for regular speed and M3 System clock / 8 for high speed
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing conventional 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
- LIN protocol support
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level
  - Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

21.2 Block Diagram

Figure 21-1 shows the UART block diagram.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 21-1. UART Module Block Diagram



21.3 Functional Description

Each 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. 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.

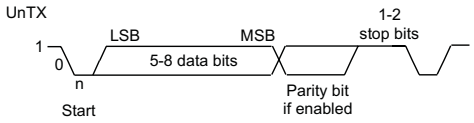
21.3.1 Transmit/Receive Logic

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 21-2 for details.

Copyright © 2012–2019, Texas Instruments Incorporated

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 21-2. UART Character Frame



21.3.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the UART Integer Baud-Rate Divisor (UARTIBRD) register and the 6-bit fractional part is loaded with the UART Fractional Baud-Rate Divisor (UARTFBRD) register. The baud-rate divisor (BRD) has the following relationship to the system clock (where BRDI is the integer part of the BRD and BRDF is the fractional part, separated by a decimal place).

BRD = BRDI + BRDF = UARTSysClk / (ClkDiv * Baud Rate)

where UARTSysClk is the system clock connected to the UART, and ClkDiv is either 16 (if HSE in UARTCTL is clear) or 8 (if HSE is set).

The 6-bit fractional number (that is to be loaded into the DIVFRAC bit field in the UARTFBRD register) can be calculated by taking the fractional part of the baud-rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors:

UARTFBRD[DIVFRAC] = integer(BRDF * 64 + 0.5)

The UART generates an internal baud-rate reference clock at 8x or 16x the baud-rate (referred to as Baud8 and Baud16, depending on the setting of the HSE bit (bit 5) in UARTCTL). This reference clock is divided by 8 or 16 to generate the transmit clock, and is used for error detection during receive operations.

Along with the UART Line Control, High Byte (UARTLCRH) register, 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

21.3.3 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 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.

Copyright © 2012–2019, Texas Instruments Incorporated

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 Section 21.3.1.

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.

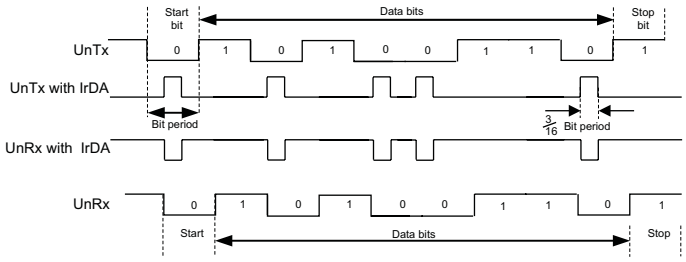
21.3.4 Serial IR (SIR)

The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream and a half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output and decoded input to the UART. When enabled, the SIR block uses the UnTx and UnRx pins for the SIR protocol. These signals should be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception.The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as a high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW and driving the UART input pin LOW.
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated IrLPBaud16 signal (1.63 µs, assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the UARTCR register. See the UARTILPR register for more information on IrDA low-power pulse-duration configuration.

Figure 21-3 shows the UART transmit and receive signals, with and without IrDA modulation.

Figure 21-3. 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.

Copyright © 2012–2019, Texas Instruments Incorporated

21.3.5 ISO 7816 Support

The UART offers basic support to allow communication with an ISO 7816 smartcard. When bit 3 (SMART) of the UARTCTL register is set, the UnTx signal is used as a bit clock, and the UnRx signal is used as the half-duplex communication line connected to the smartcard. A GPIO signal can be used to generate the reset signal to the smartcard. The remaining smartcard signals should be provided by the system design.

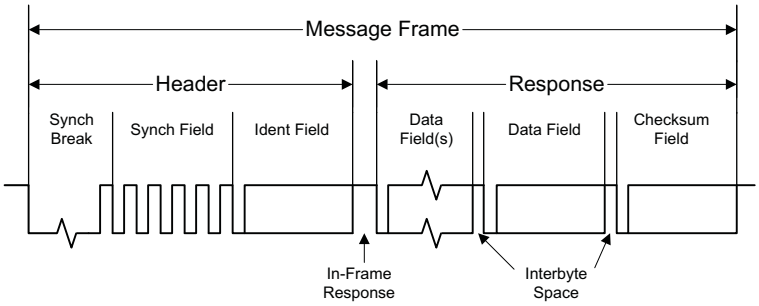
When using ISO 7816 mode, the UARTLCRH register must be set to transmit 8-bit words (WLEN bits 6:5 configured to 0x3) with EVEN parity (PEN set and EPS set). In this mode, the UART automatically uses 2 stop bits, and the STP2 bit of the UARTLCRH register is ignored.

If a parity error is detected during transmission, UnRx is pulled Low during the second stop bit. In this case, the UART aborts the transmission, flushes the transmit FIFO and discards any data it contains, and raises a parity error interrupt, allowing software to detect the problem and initiate retransmission of the affected data. Note that the UART does not support automatic retransmission in this case.

21.3.6 LIN Support

The UART module offers hardware support for the LIN protocol as either a master or a slave. The LIN mode is enabled by setting the LIN bit in the UARTCTL register. A LIN message is identified by the use of a Sync Break at the beginning of the message. The Sync Break is a transmission of a series of 0s. The Sync Break is followed by the Sync data field (0x55). Figure 21-4 illustrates the structure of a LIN message.

Figure 21-4. LIN Message



The UART should be configured as followed to operate in LIN mode:

- Configure the UART for 1 start bit, 8 data bits, no parity, and 1 stop bit. Enable the Transmit FIFO.
- Set the LIN bit in the UARTCTL register.

When preparing to send a LIN message, the TXFIFO should contain the Sync data (0x55) at FIFO location 0 and the Identifier data at location 1, followed by the data to be transmitted, and with the checksum in the final FIFO entry.

21.3.6.1 LIN Master

The UART is enabled to be the LIN master by setting the MASTER bit in the UARTLCTL register. The length of the Sync Break is programmable using the BLEN field in the UARTLCTL register and can be 13-16 bits (baud clock cycles).

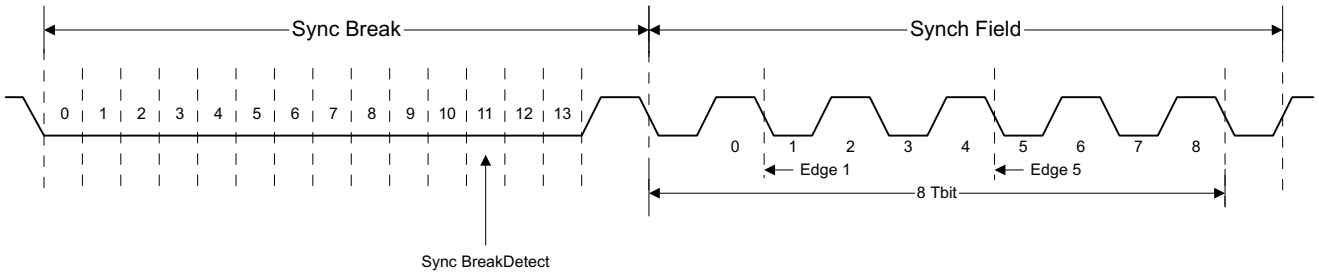
21.3.6.2 LIN Slave

The LIN UART slave is required to adjust its baud rate to that of the LIN master. In slave mode, the LIN UART recognizes the Sync Break, which must be at least 13 bits in duration. A timer is provided to capture timing data on the 1st and 5th falling edges of the Sync field so that the baud rate can be adjusted to match the master.

Copyright © 2012–2019, Texas Instruments Incorporated

After detecting a Sync Break, the UART waits for the synchronization field. The first falling edge generates an interrupt using the LME1RIS bit in the UARTRIS register, and the timer value is captured and stored in the UARTLSS register (T1). On the fifth falling edge, a second interrupt is generated using the LME5RIS bit in the UARTRIS register, and the timer value is captured again (T2). The actual baud rate can be calculated using (T2-T1)/8, and the local baud rate should be adjusted as needed. Figure 21-5 illustrates the synchronization field.

Figure 21-5. LIN Synchronization Field



21.3.7 FIFO Operation

The UART has two 16-entry FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the UART Data (UARTDR) register. 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 the UARTLCRH register.

FIFO status can be monitored via the UART Flag (UARTFR) register and the UART Receive Status (UARTRSR) register. Hardware monitors empty, full and overrun conditions. The UARTFR register contains empty and full flags (TXFE, TXFF, RXFE, and RXFF bits), and the UARTRSR register shows overrun status via the OE bit.

The trigger points at which the FIFOs generate interrupts is controlled via the UART Interrupt FIFO Level Select (UARTIFLS) register. Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include 1/8, 1/4, 1/2, 3/4, and 7/8. For example, if the 1/4 option is selected for the receive FIFO, the UART generates a receive interrupt after 4 data bytes are received. Out of reset, both FIFOs are configured to trigger an interrupt at the 1/2 mark.

21.3.8 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.

The interrupt events that can trigger a controller-level interrupt are defined in the UART Interrupt Mask (UARTIM) register by setting the corresponding IM bits. If interrupts are not used, the raw interrupt status is always visible via the UART Raw Interrupt Status (UARTRIS).

Copyright © 2012–2019, Texas Instruments Incorporated

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.

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the UARTICR register.

21.3.9 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. In loopback mode, data transmitted on the UnTx output is received on the UnRx input.

21.3.10 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 Î¼DMA controller depending on how the DMA channel is configured.

To enable DMA operation for the receive channel, set the RXDMAE bit of the DMA Control (UARTDMACTL) register. To enable DMA operation for the transmit channel, set the TXDMAE bit of the UARTDMACTL register. The UART can also be configured to stop using DMA for the receive channel if a receive error occurs. If the DMAERR bit of the UARTDMACR register is set and a receive error occurs, the DMA receive requests are automatically disabled. This error condition can be cleared by clearing the appropriate UART error interrupt.

If DMA is enabled, then the µDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the UART interrupt vector. Therefore, if interrupts are used for UART operation and DMA is enabled, the UART interrupt handler must be designed to handle the µDMA completion interrupt.

See the *Micro Direct Memory Access (µDMA*) chapter for more details about programming the µDMA controller.

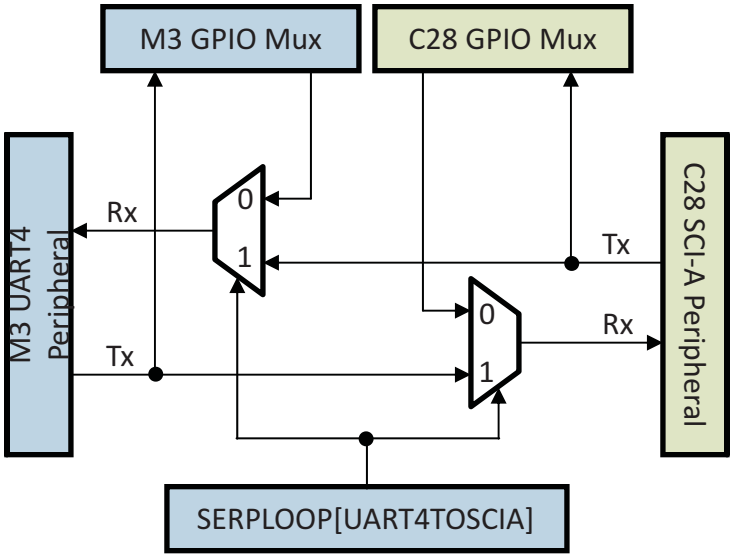
21.4 M3 UART4 to C28 SCI-A Internal Loopback

The M3 UART4 peripheral can be internally connected to the C28 SCI-A peripheral. External GPIO pins are not used when the loopback feature is enabled and can be used for other functions.

Figure 21-6 illustrates the loopback connections between the M3 UART4 and C28 SCI-A. The M3 UART4Rx is connected to C28 SCI-ATx and the M3 UART4Tx is connected to C28 SCI-ARx when loopback is enabled. Loopback is enabled by setting the UART4TOSCIA bit in the M3 SERPLOOP register. For a complete description of the SERPLOOP register, refer to the *System Control and Interrupt* chapter.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 21-6. UART and SCI Connections for Loopback Mode



21.4.1 Loopback Initialization and Configuration

To enable M3 UART4 to C28 SCI-A loopback, follow these steps:

1. Enable and configure the M3 UART4 module by following the steps outlined in Section 21.5.
2. Enable internal loopback mode by setting SERPLOOP[UART4TOSCIA] = 1 on the M3 subsystem.
3. Enable and configure the C28 SCI-A module as described in the Enhanced SCI Module Overview section of the *C28 Serial Communications Interface (SCI)* chapter.

To disable loopback between the M3 and C28, set SERPLOOP[UART4TOSCIA] = 0.

21.5 Initialization and Configuration

To enable and initialize the UART, the following steps are necessary:

- The peripheral clock must be enabled by setting the UART0, UART1, or UART2 bits in the RCGC1 register (see the *System Control and Interrupts* chapter).
- The clock to the appropriate GPIO module must be enabled via the RCGC2 register in the System Control module (see the *System Control and Interrupts* chapter).
- Set the GPIO AFSEL bits for the appropriate pins (see the GPIOs chapter). To determine which GPIOs to configure, see the *GPIOs* chapter.
- Configure the GPIO current level and/or slew rate as specified for the mode selected (see the *GPIOs* chapter).
- Configure the PMCn fields in the GPIOPCTL register to assign the UART signals to the appropriate pins (see the *GPIOs* chapter).

To use the UARTs, the peripheral clock must be enabled by setting the UART0, UART1, or UART2 bits in the RCGC1 register (see the *System Control and Interrupts* chapter). In addition, the clock to the appropriate GPIO module must be enabled via the RCGC2 register in the *System Control and Interrupts* chapter. To find out which GPIO port to enable, refer to the *GPIOs* chapter.

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

Copyright © 2012–2019, Texas Instruments Incorporated

- 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 Section 21.3.2, the BRD can be calculated:

BRD = 20,000,000 / (16 * 115,200) = 10.8507

which means that the DIVINT field of the UARTIBRD register should be set to 10 decimal or 0xA. The value to be loaded into the UARTFBRD register 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:

- Disable the UART by clearing the UARTEN bit in the UARTCTL register.
- Write the integer portion of the BRD to the UARTIBRD register.
- Write the fractional portion of the BRD to the UARTFBRD register.
- Write the desired serial parameters to the UARTLCRH register (in this case, a value of 0x0000.0060).
- Optionally, configure the µDMA channel (see the *Micro Direct Memory Access (µDMA)* chapter) and enable the DMA option(s) in the UARTDMACTL register.
- Enable the UART by setting the UARTEN bit in the UARTCTL register.

21.6 Register Map

Figure 21-7 lists the UART registers and their offsets.

Note that the UART module clock must be enabled before the registers can be programmed. See the Run Mode Clock Gating Control Register 1 (RCGC1) in the *System Control and Interrupts* chapter. . There must be a delay of three system clocks after the UART module clock is enabled before any UART module registers are accessed.

NOTE: The UART must be disabled (see the UARTEN bit in the UARTCTL register 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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 21-1. Register Map

Offset	Name	Type	Reset	Description
0x000	UARTDR	R/W	0x0000.0000	UART Data
0x004	UARTRSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear
0x018	UARTFR	RO	0x0000.0090	UART Flag
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control
0x030	UARTCTL	R/W	0x0000.0300	UART Control
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear
0x048	UARTDMACTL	R/W	0x0000.0000	UART DMA Control
0x090	UARTLCTL	R/W	0x0000.0000	UART LIN Control
0x094	UARTLSS	RO	0x0000.0000	UART LIN Snap Shot
0x098	UARTLTIM	RO	0x0000.0000	UART LIN Timer
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7
0xFE0	UARTPeriphID0	RO	0x0000.0060	UART Peripheral Identification 0
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3

21.7 Register Descriptions

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

21.7.1 UART Data Register (UARTDR), offset 0x000

UARTDR is the data register (the interface to the FIFOs).

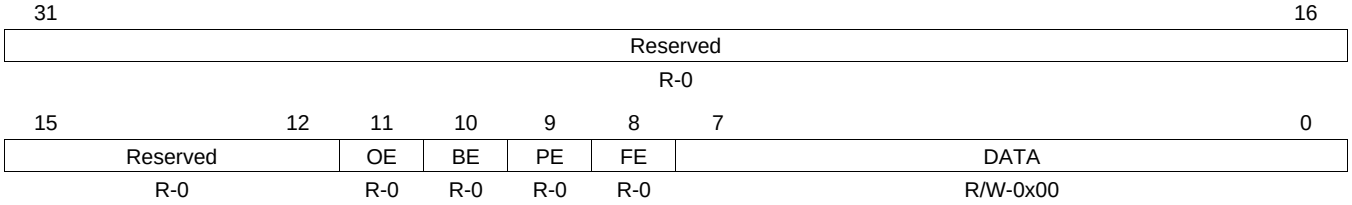
NOTE: This register is read-sensitive.

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.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 21-7. UART Data Register (UARTDR)



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 21-2. UART Data Register (UARTDR) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved		Reseved
11	OE		UART Overrun Error
		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.
10	BE		UART Break Error 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.
		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).
9	PE		UART Parity Error In FIFO mode, this error is associated with the character at the top of the FIFO.
		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.
8	FE		UART Framing Error
		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		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.

21.7.2 UART Receive Status/Error Clear Register (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

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 21-8. UART Receive Status Register (UARTRSR/UARTECR)

31					16
Reserved					
R-0					

15	4	3	2	1	0
Reserved		OE	BE	PE	FE
R-0		R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-3. UART Receive Status Register (UARTRSR/UARTECR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reserved
3	OE		UART Overrun Error 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.
		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.
2	BE		UART Break Error 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.
		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).
1	PE		UART Parity Error This bit is cleared to 0 by a write to UARTECR.
		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.
0	FE		UART Framing Error 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.
		0	No framing error has occurred.
		1	The received character does not have a valid stop bit (a valid stop bit is 1).

Copyright © 2012–2019, Texas Instruments Incorporated

WRITE-ONLY Error Clear Register

Figure 21-9. UART Receive Status/Error Clear Register (UARTRSR/UARTECR)

31	8	7	0
Reserved		DATA	
W-0		W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-4. UART Error Clear (UARTECR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	DATA		Error Clear A write to this register of any data clears the framing, parity, break, and overrun flags.

21.7.3 UART Flag Register (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.

Figure 21-10. UART Flag Register (UARTFR)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	0
Reserved		TXFE	RXFF	TXFF	RXFE	BUSY	Reserved	
R-0		R-1	R-0	R-0	R-1	R-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-5. UART Flag Register (UARTFR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7	TXFE		UART Transmit FIFO Empty The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register.
		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		UART Receive FIFO Full The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register.
		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.
5	TXFF		UART Transmit FIFO Full The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register.
		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		UART Receive FIFO Empty The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register.
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 21-5. UART Flag Register (UARTFR) Field Descriptions (continued)

Bit	Field	Value	Description
3	BUSY		UART Busy This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).
		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.
2-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.4 UART IrDA Low-Power Register (UARTILPR), offset 0x020

The UARTILPR register stores the 8-bit low-power counter divisor value used to derive the low-power SIR pulse width clock by dividing down the system clock (SysClk). All the bits are cleared when reset.

The internal IrLPBaud16 clock is generated by dividing down SysClk according to the low-power divisor value written to UARTILPR. The duration of SIR pulses generated when low-power mode is enabled is three times the period of the IrLPBaud16 clock. The low-power divisor value is calculated as follows:

ILPDVSR = SysClk / $F_{IrLPBaud16}$

where $F_{IrLPBaud16}$ is nominally 1.8432 MHz.

The divisor must be programmed such that 1.42 MHz < $F_{IrLPBaud16}$ < 2.12 MHz, resulting in a low-power pulse duration of 1.41-2.11 µs (three times the period of IrLPBaud16). The minimum frequency of IrLPBaud16 ensures that pulses less than one period of IrLPBaud16 are rejected, but pulses greater than 1.4 µs are accepted as valid pulses.

NOTE: Zero is an illegal value. Programming a zero value results in no IrLPBaud16 pulses being generated.

Figure 21-11. UART IrDA Low-Power Register (UARTILPR)

31	8	7	0
Reseved		ILPDVSR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-6. UART IrDA Low-Power Register (UARTILPR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	ILPDVSR		IrDA Low-Power Divisor This field contains the 8-bit low-power divisor value.

21.7.5 UART Integer Baud-Rate Divisor Register (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 Section 21.3.2 for configuration details.

Figure 21-12. UART Integer Baud-Rate Divisor Register (UARTIBRD)

31	16	15	0
Reserved		DIVINT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-7. UART Integer Baud-Rate Divisor (UARTIBRD) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	DIVINT		Integer Baud-Rate Divisor

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.6 UART Fractional Baud-Rate Divisor Register (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 Section 21.3.2 for configuration details.

Figure 21-13. UART Fractional Baud-Rate Divisor Register (UARTFBRD)

31	6	5	0
Reserved		DIVFRAC	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-8. UART Fractional Baud-Rate Divisor (UARTFBRD) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-0	DIVFRAC		Fractional Baud-Rate Divisor

21.7.7 UART Line Control Register (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.

Figure 21-14. UART Line Control Register (UARTLCRH)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		SPS	WLEN		FEN	STP2	EPS	PEN	BRK
R-0		R/W-0	R/W-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-9. UART Line Control Register (UARTLCRH) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7	SPS		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		UART Word Length The bits indicate the number of data bits transmitted or received in a frame as follows:
		0x0	5 bits (default)
		0x1	6 bits
		0x2	7 bits
		0x3	8 bits
4	FEN		UART Enable FIFOs
		0	The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.
		1	The transmit and receive FIFObuffers are enabled (FIFOmode).

Copyright © 2012–2019, Texas Instruments Incorporated

Table 21-9. UART Line Control Register (UARTLCRH) Field Descriptions (continued)

Bit	Field	Value	Description
3	STP2		UART Two Stop Bits Select
		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		UART Even Parity Select This bit has no effect when parity is disabled by the PEN bit.
		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.
1	PEN		UART Parity Enable
		0	Parity is disabled and no parity bit is added to the data frame.
		1	Parity checking and generation is enabled.
0	BRK		UART Send Break
		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).

21.7.8 UART Control Register (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.

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.

Figure 21-15. UART Control (UARTCTL) Register

31						10	9	8
Reserved							RXE	TXE
R-0							R/W-1	R/W-1
7	6	5	4	3	2	1	0	
LBE	LIN	HSE	EOT	SMART	SIRLP	SIREN	UARTEN	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 21-10. UART Control (UARTCTL) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Reserved
9	RXE		UART Receive Enable 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.
		0	The receive section of the UART is disabled.
		1	The receive section of the UART is enabled.
8	TXE		UART Transmit Enable 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.
		0	The transmit section of the UART is disabled.
		1	The transmit section of the UART is enabled.
7	LBE		UART Loop Back Enable
		0	Normal operation.
		1	The UnTx path is fed through the UnRx path.
6	LIN		LIN Mode Enable
		0	Normal operation.
		1	The UART operates in LIN mode.
5	HSE		High-Speed Enable **Note:** System clock used is also dependent on the baud-rate divisor configuration (see Section 21.3.2).
		0	The UART is clocked using the system clock divided by 16.
		1	The UART is clocked using the system clock divided by 8.
4	EOT		End of Transmission This bit determines the behavior of the TXRIS bit in the UARTRIS register.
		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		ISO 7816 Smart Card Support 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.
		0	Normal operation.
		1	The UART operates in Smart Card mode.
2	SIRLP		UART SIR Low-Power Mode This bit selects the IrDA encoding mode. Setting this bit uses less power, but might reduce transmission distances. See the UARTILPR register.
		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.
1	SIREN		UART SIR Enable
		0	Normal operation.
		1	The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.
0	UARTEN		UART Enable If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.
		0	The UART is disabled.
		1	The UART is enabled.

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.9 UART Interrupt FIFO Level Select (UARTIFLS) Register, 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.

Figure 21-16. UART Interrupt FIFO Level Select (UARTIFLS) Register

31	16
Reserved	
R-0	

15	6	5	3	2	0
Reserved		RXIFLSEL		TXIFLSEL	
R-0		R/W-2		R/W-2	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-11. UART Interrupt FIFO Level Select (UARTIFLS) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-3	RXIFLSEL		UART Receive Interrupt FIFO Level Select The trigger points for the receive interrupt are as follows:
		0x0	RX FIFO ≥ 1/8 full
		0x1	RX FIFO ≥ 1/4 full
		0x2	RX FIFO ≥ 1/2 full (default)
		0x3	RX FIFO ≥ 3/4 full
		0x4	RX FIFO ≥ 7/8 full
		0x5–0x7	Reserved
2-0	TXIFLSEL		UART Transmit Interrupt FIFO Level Select **Note:** If the EOT bit in UARTCTL is set, 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. The trigger points for the transmit interrupt are as follows:
		0x0	TX FIFO ≤ 1/8 full
		0x1	TX FIFO ≤ 1/4 full
		0x2	TX FIFO ≤ 1/2 full (default)
		0x3	TX FIFO ≤ 3/4 full
		0x4	TX FIFO ≤ 7/8 full
		0x5–0x7	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.10 UART Interrupt Mask (UARTIM) Register, 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.

Figure 21-17. UART Interrupt Mask (UARTIM) Register

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved							
R-0							
15	14	13	12	11	10	9	8
LME5IM	LME1IM	LMSBIM	Reserved		OEIM	BEIM	PEIM
R/W-0	R/W-0	R/W-0	R-0		R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
FEIM	RTIM	TXIM	RXIM	Reserved			
R/W-0	R/W-0	R/W-0	R/W-0	R-0			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-12. UART Interrupt Mask (UARTIM) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	LME5IM		LIN Mode Edge 5 Interrupt Mask
		0	The LME5RIS interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the LME5RIS bit in the UARTRIS register is set.
14	LME1IM		LIN Mode Edge 1 Interrupt Mask
		0	The LME1RIS interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the LME1RIS bit in the UARTRIS register is set.
13	LMSBIM		LIN Mode Sync Break Interrupt Mask
		0	The LMSBRIS interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the LMSBRIS bit in the UARTRIS register is set.
12-11	Reserved		Reserved
10	OEIM		UART Overrun Error Interrupt Mask
		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.
9	BEIM		UART Break Error Interrupt Mask
		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		UART Parity Error Interrupt Mask
		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		UART Framing Error Interrupt Mask
		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		UART Receive Time-Out Interrupt Mask
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 21-12. UART Interrupt Mask (UARTIM) Register Field Descriptions (continued)

Bit	Field	Value	Description
5	TXIM		UART Transmit Interrupt Mask
		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		UART Receive Interrupt Mask
		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.
3-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.11 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.

Figure 21-18. UART Raw Interrupt Status (UARTRIS) Register

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved							
R-0							
15	14	13	12	11	10	9	8
LME5RIS	LME1RIS	LMSBRIS	Reserved		OERIS	BERIS	PERIS
R-0	R-0	R-0			R-0	R-0	R-0
7	6	5	4	3			0
FERIS	RTRIS	TXRIS	RXRIS	Reserved			
R-0	R-0	R-0	R-0	R-0			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-13. UART Raw Interrupt Status (UARTRIS) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	LME5RIS		LIN Mode Edge 5 Raw Interrupt Status This bit is cleared by writing a 1 to the LME5IC bit in the UARTICR register.
		0	No interrupt
		1	The timer value at the 5th falling edge of the LIN Sync Field has been captured.
14	LME1RIS		LIN Mode Edge 1 Raw Interrupt Status This bit is cleared by writing a 1 to the LME1IC bit in the UARTICR register.
		0	No interrupt
		1	The timer value at the 1st falling edge of the LIN Sync Field has been captured.
13	LMSBRIS		LIN Mode Sync Break Raw Interrupt Status This bit is cleared by writing a 1 to the LMSBIC bit in the UARTICR register.
		0	No interrupt
		1	A LIN Sync Break has been detected.
12-11	Reserved		Reserved
10	OERIS		UART Overrun Error Raw Interrupt Status This bit is cleared by writing a 1 to the OEIC bit in the UARTICR register.
		0	No interrupt
		1	An overrun error has occurred.
9	BERIS		UART Break Error Raw Interrupt Status This bit is cleared by writing a 1 to the BEIC bit in the UARTICR register.
		0	No interrupt
		1	A break error has occurred.
8	PERIS		UART Parity Error Raw Interrupt Status This bit is cleared by writing a 1 to the PEIC bit in the UARTICR register.
		0	No interrupt
		1	A parity error has occurred.
7	FERIS		UART Framing Error Raw Interrupt Status This bit is cleared by writing a 1 to the FEIC bit in the UARTICR register.
		0	No interrupt
		1	A framing error has occurred.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 21-13. UART Raw Interrupt Status (UARTRIS) Register Field Descriptions (continued)

Bit	Field	Value	Description
6	RTRIS		UART Receive Time-Out Raw Interrupt Status This bit is cleared by writing a 1 to the RTIC bit in the UARTICR register.
		0	No interrupt
		1	A receive time out has occurred.
5	TXRIS		UART Transmit Raw Interrupt Status This bit is cleared by writing a 1 to the TXIC bit in the UARTICR register.
		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.
4	RXRIS		UART Receive Raw Interrupt Status This bit is cleared by writing a 1 to the RXIC bit in the UARTICR register.
		0	No interrupt
		1	The receive FIFO level has passed through the condition defined in the UARTIFLS register.
3-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.12 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.

Figure 21-19. UART Masked Interrupt Status (UARTMIS) Register

31	30	29	28	27	26	25	24
Reserved							
R-0							
23	22	21	20	19	18	17	16
Reserved							
R-0							
15	14	13	12	11	10	9	8
LME5MIS	LME1MIS	LMSBMIS	Reserved		OEMIS	BEMIS	PEMIS
R-0	R-0	R-0	R-0		R-0	R-0	R-0
7	6	5	4	3	2	1	0
FEMIS	RTMIS	TXMIS	RXMIS	Reserved			
R-0	R-0	R-0	R-0	R-0			

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-14. UART Masked Interrupt Status (UARTMIS) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	LME5MIS		LIN Mode Edge 5 Masked Interrupt Status This bit is cleared by writing a 1 to the LME5IC bit in the UARTICR register.
		0	An interrupt has not occurred or is masked.
		1	An unmasked interrupt was signaled due to the 5th falling edge of the LIN Sync Field.
14	LME1MIS		LIN Mode Edge 1 Masked Interrupt Status This bit is cleared by writing a 1 to the LME1IC bit in the UARTICR register.
		0	An interrupt has not occurred or is masked.
		1	An unmasked interrupt was signaled due to the 1st falling edge of the LIN Sync Field.
13	LMSBMIS		LIN Mode Sync Break Masked Interrupt Status This bit is cleared by writing a 1 to the LMSBIC bit in the UARTICR register.
		0	An interrupt has not occurred or is masked.
		1	An unmasked interrupt was signaled due to the receipt of a LIN Sync Break.
12-11	Reserved		Reserved
10	OEMIS		UART Overrun Error Masked Interrupt Status This bit is cleared by writing a 1 to the OEIC bit in the UARTICR register.
		0	An interrupt has not occurred or is masked.
		1	An unmasked interrupt was signaled due to an overrun error.
9	BEMIS		UART Break Error Masked Interrupt Status This bit is cleared by writing a 1 to the BEIC bit in the UARTICR register.
		0	An interrupt has not occurred or is masked.
		1	An unmasked interrupt was signaled due to a break error.
8	PEMIS		UART Parity Error Masked Interrupt Status This bit is cleared by writing a 1 to the PEIC bit in the UARTICR register.
		0	An interrupt has not occurred or is masked.
		1	An unmasked interrupt was signaled due to a parity error.
7	FEMIS		UART Framing Error Masked Interrupt Status This bit is cleared by writing a 1 to the FEIC bit in the UARTICR register.
		0	An interrupt has not occurred or is masked.
		1	An unmasked interrupt was signaled due to a framing error.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 21-14. UART Masked Interrupt Status (UARTMIS) Register Field Descriptions (continued)

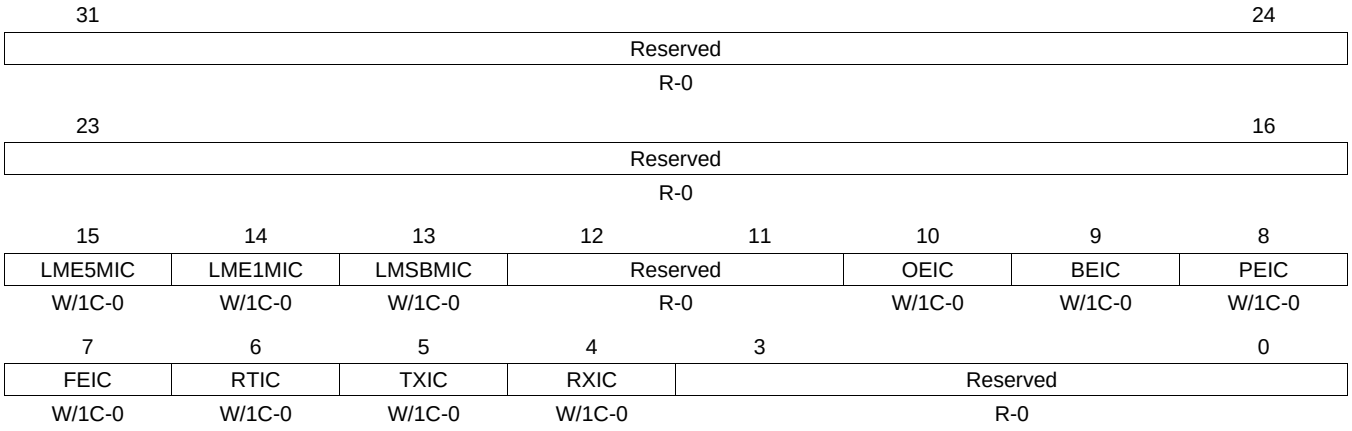
Bit	Field	Value	Description
6	RTMIS		UART Receive Time-Out Masked Interrupt Status This bit is cleared by writing a 1 to the RTIC bit in the UARTICR register.
		0	An interrupt has not occurred or is masked.
		1	An unmasked interrupt was signaled due to a receive time out.
5	TXMIS		UART Transmit Masked Interrupt Status This bit is cleared by writing a 1 to the TXIC bit in the UARTICR register.
		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).
4	RXMIS		UART Receive Masked Interrupt Status This bit is cleared by writing a 1 to the RXIC bit in the UARTICR register.
		0	An unmasked interrupt was signaled due to passing through the specified receive FIFO level. An interrupt has not occurred or is masked.
		1	
3-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.13 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.

Figure 21-20. UART Interrupt Clear (UARTICR) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-15. UART Interrupt Clear (UARTICR) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	LME5MIC		LIN Mode Edge 5 Interrupt Clear Writing a 1 to this bit clears the LME5RIS bit in the UARTRIS register and the LME5MIS bit in the UARTMIS register.
14	LME1MIC		LIN Mode Edge 1 Interrupt Clear Writing a 1 to this bit clears the LME1RIS bit in the UARTRIS register and the LME1MIS bit in the UARTMIS register.
13	LMSBMIC		LIN Mode Sync Break Interrupt Clear Writing a 1 to this bit clears the LMSBRIS bit in the UARTRIS register and the LMSBMIS bit in the UARTMIS register.
12-11	Reserved		Reserved
10	OEIC		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		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		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		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		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.
5	TXIC		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		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.
3-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.14 UART DMA Control (UARTDMACTL), offset 0x048

The UARTDMACTL register is the DMA control register.

Figure 21-21. UART DMA Control (UARTDMACTL) Register

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		DMAERR	TXDMAE	RXDMAE
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 21-16. UART DMA Control (UARTDMACTL) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	DMAERR		DMA on Error
		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		Transmit DMA Enable
		0	µDMA for the transmit FIFO is disabled.
		1	µDMA for the transmit FIFO is enabled.
0	RXDMAE		Receive DMA Enable
		0	µDMA for the receive FIFO is disabled.
		1	µDMA for the receive FIFO is enabled.

21.7.15 UART LIN Control (UARTLCTL), offset 0x090

The UARTLCTL register is the configures the operation of the UART when in LIN mode.

Figure 21-22. UART LIN Control (UARTLCTL) Register

31	16
Reserved	
R-0	

15	6	5	4	3	1	0
Reserved		BLEN		Reserved		MASTER
R-0		R/W-0		R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 21-17. UART LIN Control (UARTLCTL) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-4	BLEN		Sync Break Length
		0x3	Sync break length is 16T bits
		0x2	Sync break length is 15T bits
		0x1	Sync break length is 14T bits
		0x0	Sync break length is 13T bits (default)
3-1	Reserved		Reserved
0	MASTER		LIN Master Enable
		0	The UART operates as a LIN slave.
		1	The UART operates as a LIN master.

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.16 UART LIN Snap Shot (UARTLSS), offset 0x094

The UARTLSS register captures the free-running timer value when either the Sync Edge 1 or the Sync Edge 5 is detected in LIN mode.

Figure 21-23. UART LIN Snap Shot (UARTLSS) Register

31	16	15	0
Reserved		TSS	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-18. UART LIN Snap Shot (UARTLSS) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	TSS		Timer Snap Shot This field contains the value of the free-running timer when either the Sync Edge 5 or the Sync Edge 1 was detected.

21.7.17 UART LIN Timer (UARTLTIM), offset 0x098

The UARTLTIM register contains the current timer value for the free-running timer that is used to calculate the baud rate when in LIN slave mode. The value in this register is used along with the value in the UART LIN Snap Shot (UARTLSS) register to adjust the baud rate to match that of the master.

Figure 21-24. UART LIN Timer (UARTLTIM) Register

31	16	15	0
Reserved		TIMER	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-19. UART LIN Timer (UARTLTIM) Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15-0	TIMER		Timer Value This field contains the value of the free-running timer.

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.18 UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-25. UART Peripheral Identification 4 (UARTPeriphID4) Register

31	8	7	0
Reserved		PID4	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-20. UART Peripheral Identification 4 (UARTPeriphID4) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID4		UART Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

21.7.19 UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-26. UART Peripheral Identification 5 (UARTPeriphID5) Register

31	8	7	0
Reserved		PID5	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-21. UART Peripheral Identification 5 (UARTPeriphID5) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID5		UART Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

21.7.20 UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-27. UART Peripheral Identification 6 (UARTPeriphID6) Register

31	8	7	0
Reserved		PID6	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-22. UART Peripheral Identification 6 (UARTPeriphID6) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID6		UART Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.21 UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-28. UART Peripheral Identification 7 (UARTPeriphID7) Register

31	8	7	0
Reserved		PID7	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-23. UART Peripheral Identification 7 (UARTPeriphID7) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID7		UART Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

21.7.22 UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-29. UART Peripheral Identification 0 (UARTPeriphID0) Register

31	8	7	0
Reserved		PID0	
R-0		R-0x60	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-24. UART Peripheral Identification 0 (UARTPeriphID0) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID0		UART Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

21.7.23 UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-30. UART Peripheral Identification 1 (UARTPeriphID1) Register

31	8	7	0
Reserved		PID1	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-25. UART Peripheral Identification 1 (UARTPeriphID1) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID1		UART Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.24 UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-31. UART Peripheral Identification 2 (UARTPeriphID2) Register

31	8	7	0
Reserved		PID2	
R-0		R-0x18	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-26. UART Peripheral Identification 2 (UARTPeriphID2) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID2		UART Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

21.7.25 UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The UARTPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-32. UART Peripheral Identification 3 (UARTPeriphID3) Register

31	8	7	0
Reserved		PID3	
R-0		R-0x01	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-27. UART Peripheral Identification 3 (UARTPeriphID3) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	PID3		UART Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

21.7.26 UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

The UARTPCellIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-33. UART PrimeCell Identification 0 (UARTPCellID0) Register

31	8	7	0
Reserved		CID0	
R-0		R-0x0D	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-28. UART PrimeCell Identification 0 (UARTPCellID0) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID0		UART PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Copyright © 2012–2019, Texas Instruments Incorporated

21.7.27 UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

The UARTPCellIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-34. UART PrimeCell Identification 1 (UARTPCellID1) Register

31	8	7	0
Reserved		CID1	
R-0		R-0xF0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-29. UART PrimeCell Identification 1 (UARTPCellID1) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID1		UART PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

21.7.28 UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

The UARTPCellIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-35. UART PrimeCell Identification 2 (UARTPCellID2) Register

31	8	7	0
Reserved		CID2	
R-0		R-0x05	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-30. UART PrimeCell Identification 2 (UARTPCellID2) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID2		UART PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

21.7.29 UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC

The UARTPCellIDn registers are hard-coded and the fields within the registers determine the reset values.

Figure 21-36. UART PrimeCell Identification 3 (UARTPCellID3) Register

31	8	7	0
Reserved		CID3	
R-0		R-0xB1	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-31. UART PrimeCell Identification 3 (UARTPCellID3) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	CID3		UART PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

Copyright © 2012–2019, Texas Instruments Incorporated

M3 Inter-Integrated Circuit (I2C) Interface

This chapter describes the features and operation of the M3 inter-integrated circuit (I2C) module.

Topic **Page**

22.1 Introduction **1485**
22.2 I2C Block Diagram **1485**
22.3 Functional Description **1485**
22.4 Initialization and Configuration **1497**
22.5 Register Map **1498**
22.6 Register Descriptions **1499**
22.7 Register Descriptions (I2C Slave) **1508**

Copyright © 2012–2019, Texas Instruments Incorporated

22.1 Introduction

The M3 Inter-Integrated Circuit (I2C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I2C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I2C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The microcontroller includes two I2C modules, providing the ability to interact (both transmit and receive) with other I2C devices on the bus.

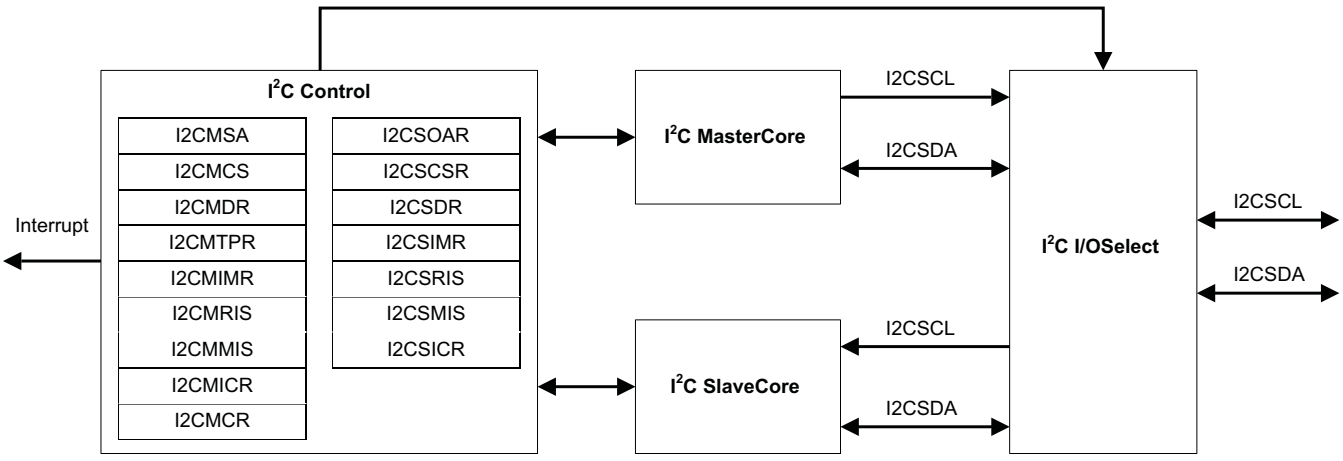
The two I2C modules include the following features:

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

22.2 I2C Block Diagram

Figure 22-1 shows the block diagram for I2C.

Figure 22-1. I2C Block Diagram



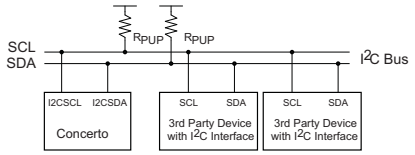
22.3 Functional Description

Each I2C module is comprised of both master and slave functions. For proper operation, the SDA and SCL pins must be configured as open-drain signals. A typical I2C bus configuration is shown in Figure 22-2.

See *Inter-Integrated Circuit (I2C) Interface* electricals in the device data manual for I2C timing diagrams.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 22-2. I2C Bus Configuration



22.3.1 I2C Bus Functional Overview

The I2C bus uses only two signals: SDA and SCL, named I2CSDA and I2CSCL. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are high.

Every transaction on the I2C bus is nine bits long, consisting of eight data bits and a single acknowledge bit. The number of bytes per transfer (defined as the time between a valid START and STOP condition, described in Section 22.3.1.1, is unrestricted, but each byte has to be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it can hold the clock line SCL Low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.

22.3.1.1 START and STOP Conditions

The protocol of the I2C bus defines two states to begin and end a transaction: START and STOP. A high-to-low transition on the SDA line while the SCL is high is defined as a START condition, and a low-to-high transition on the SDA line while SCL is high is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 22-3.

Figure 22-3. START and STOP Conditions

The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the I2C Master Slave Address (I2CMSA) register is written with the desired address, the R/S bit is cleared, and the Control register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data may be read from the I2C Master Data (I2CMDR) register. When the I2C module operates in Master receiver mode, the ACK bit is normally set causing the I2C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I2C bus controller requires no further data to be transmitted from the slave transmitter.

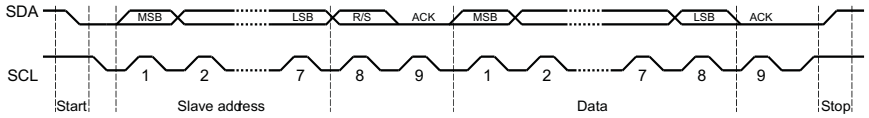
When operating in slave mode, two bits in the I2C Slave Raw Interrupt Status (I2CSRIS) register indicate detection of start and stop conditions on the bus; while two bits in the I2C Slave Masked Interrupt Status (I2CSMIS) register allow start and stop conditions to be promoted to controller interrupts (when interrupts are enabled).

22.3.1.2 Data Format with 7-Bit Address

Data transfers follow the format shown in Figure 22-4. 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.

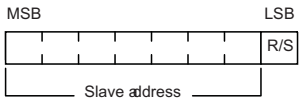
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 22-4. Complete Data Transfer with a 7-Bit Address



The first seven bits of the first byte make up the slave address (see Figure 22-5). 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 1 (one) in this position means that the master receives data from the slave.

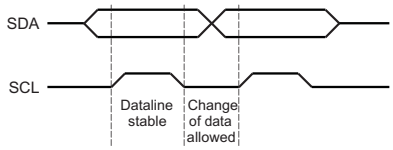
Figure 22-5. R/S Bit in First Byte



22.3.1.3 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 22-6).

Figure 22-6. Data Validity During Bit Transfer on the I2C Bus



22.3.1.4 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 Section 22.3.1.3.

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.

22.3.1.5 Arbitration

A master may start a transfer only if the bus is idle. It is 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.

22.3.2 Available Speed Modes

The I2C bus can run in either standard mode (100 kbps) or fast mode (400 kbps). The selected mode should match the speed of the other I2C devices on the bus

Copyright © 2012–2019, Texas Instruments Incorporated

22.3.2.1 Standard and Fast Modes

Standard and Fast modes are selected using a value in the I2C Master Timer Period (I2CMTPR) register that results in an SCL frequency of 100 kbps for Standard mode or 400 kbps for Fast mode.

The I2C clock rate is determined by the parameters CLK_PRD, TIMER_PRD, SCL_LP, and SCL_HP where:

CLK_PRD is the system clock period

SCL_LP is the low phase of SCL (fixed at 6)

SCL_HP is the high phase of SCL (fixed at 4)

TIMER_PRD is the programmed value in the I2CMTPR register.

The I2C clock period is calculated as follows:

$SCL\_PERIOD = 2 \times (1 + TIMER\_PRD) \times (SCL\_LP + SCL\_HP) \times CLK\_PRD$

For example:

CLK_PRD = 50 ns

TIMER_PRD = 2

SCL_LP=6

SCL_HP=4

yields a SCL frequency of:

1/SCL_PERIOD = 333 Khz

Table 22-1 gives examples of the timer periods that should be used to generate both standard and fast mode SCL frequencies based on various system clock frequencies.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 22-1. Examples of I2C Master Timer Period versus Speed Mode

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
4 MHz	0x01	100 Kbps	-	
6 MHz	0x02	100 Kbps	-	
12.5 MHz	0x06	89 Kbps	0x01	312 Kbps
16.7 MHz	0x08	93 Kbps	0x02	278 Kbps
20 MHz	0x09	100 Kbps	0x02	333 Kbps
25 MHz	0x0C	96.2 Kbps	0x03	312 Kbps
33 MHz	0x10	97.1 Kbps	0x04	330 Kbps
40 MHz	0x13	100 Kbps	0x04	400 Kbps
50 MHz	0x18	100 Kbps	0x06	357 Kbps
80 MHz	0x27	100 Kbps	0x09	400 Kbps

22.3.3 Interrupts

The I2C module can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master arbitration lost
- Master transaction error
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

The I2C master and I2C slave modules have separate interrupt signals. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

22.3.3.1 I2C Master Interrupts

The I2C master module generates an interrupt when a transaction completes (either transmit or receive), when arbitration is lost, or when an error occurs during a transaction. To enable the I2C master interrupt, software must set the IM bit in the I2C Master Interrupt Mask (I2CMIMR) register. When an interrupt condition is met, software must check the ERROR and ARBLST bits in the I2C Master Control/Status (I2CMCS) register to verify that an error didn't occur during the last transaction and to ensure that arbitration has not been lost. An error condition is asserted if the last transaction wasn't acknowledged by the slave. If an error is not detected and the master has not lost arbitration, the application can proceed with the transfer. The interrupt is cleared by writing a 1 to the IC bit in the I2C Master Interrupt Clear (I2CMICR) register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the I2C Master Raw Interrupt Status (I2CMRIS) register.

22.3.3.2 I2C Slave Interrupts

The slave module can generate an interrupt when data has been received or requested. This interrupt is enabled by setting the DATAIM bit in the I2C Slave Interrupt Mask (I2CSIMR) register. Software determines whether the module should write (transmit) or read (receive) data from the I2C Slave Data (I2CSDR) register, by checking the RREQ and TREQ bits of the I2C Slave Control/Status (I2CSCSR) register. If the slave module is in receive mode and the first byte of a transfer is received, the FBR bit is set along with the RREQ bit. The interrupt is cleared by setting the DATAIC bit in the I2C Slave Interrupt Clear (I2CSICR) register.

In addition, the slave module can generate an interrupt when a start and stop condition is detected. These interrupts are enabled by setting the STARTIM and STOPIM bits of the I2C Slave Interrupt Mask (I2CSIMR) register and cleared by writing a 1 to the STOPIC and STARTIC bits of the I2C Slave Interrupt Clear (I2CSICR) register.

Copyright © 2012–2019, Texas Instruments Incorporated

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the I2C Slave Raw Interrupt Status (I2CSRIS) register.

22.3.4 Loopback Operation

The I2C modules can be placed into an internal loopback mode for diagnostic or debug work by setting the LPBK bit in the I2C Master Configuration (I2CMCR) register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

22.3.5 Command Sequence Flow Charts

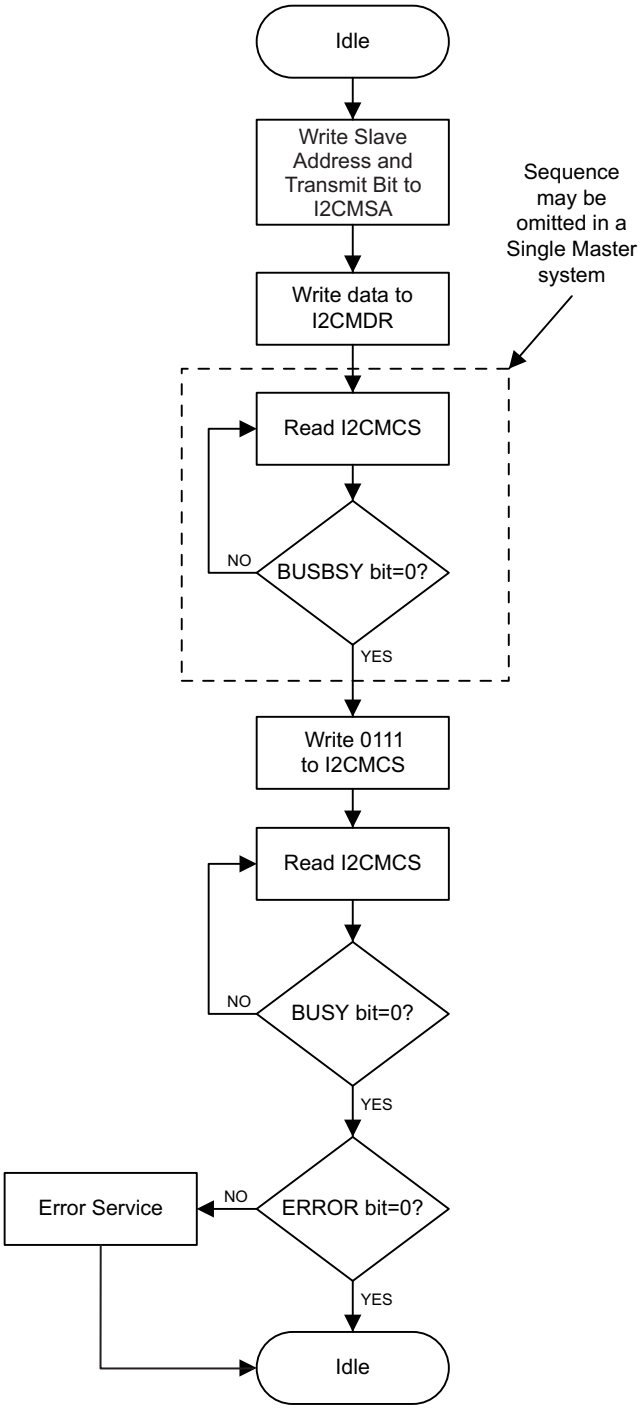
This section details the steps required to perform the various I2C transfer types in both master and slave mode.

22.3.5.1 I2C Master Command Sequences

Figure 22-7 through Figure 22-12 show the command sequences available for the I2C master.

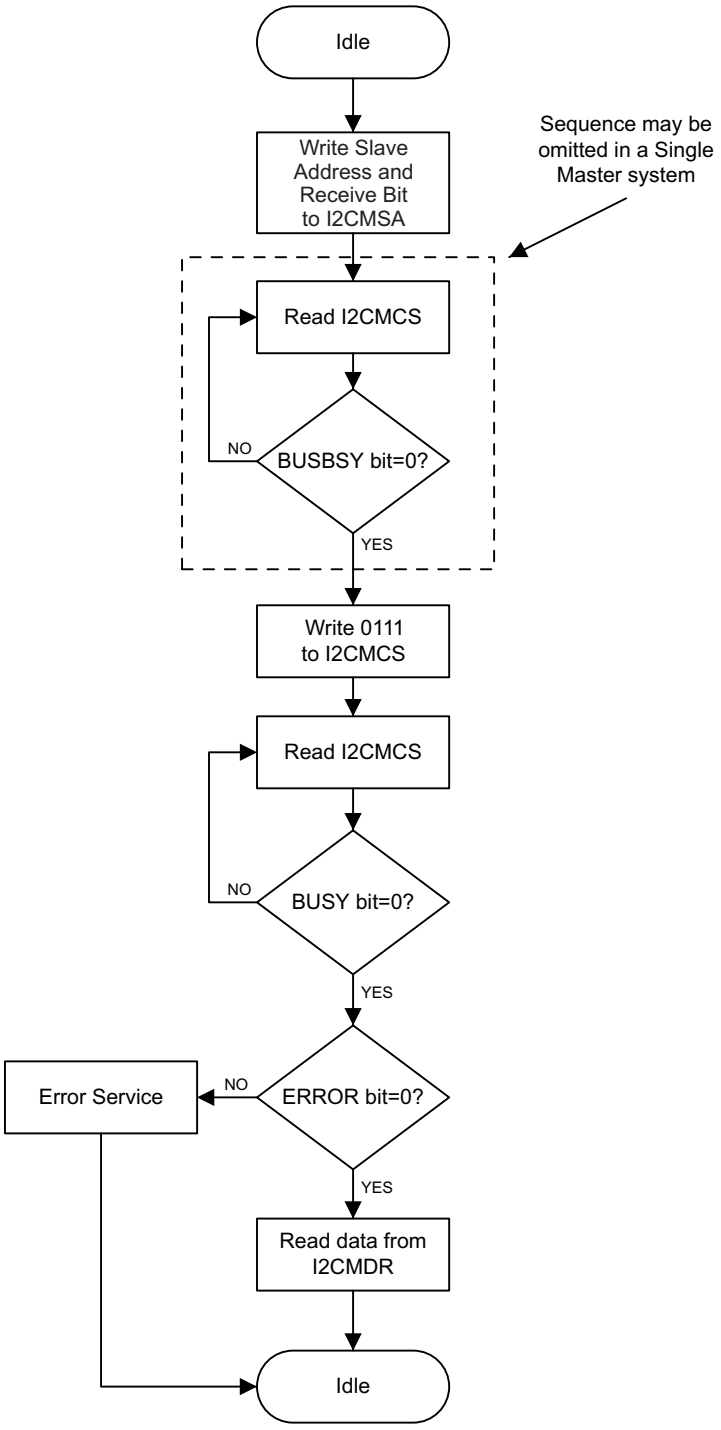
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 22-7. Master Single TRANSMIT



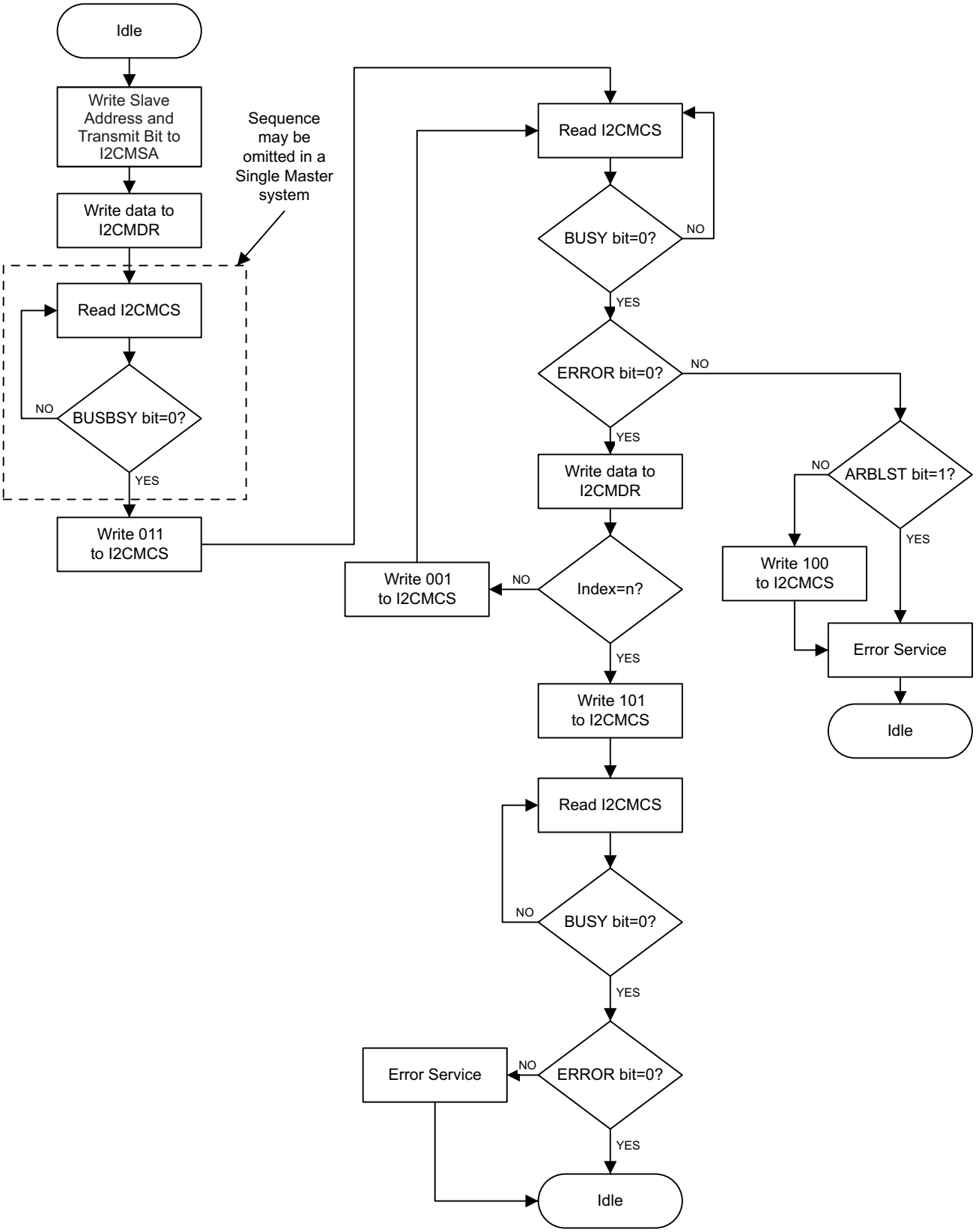
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 22-8. Master Single RECEIVE



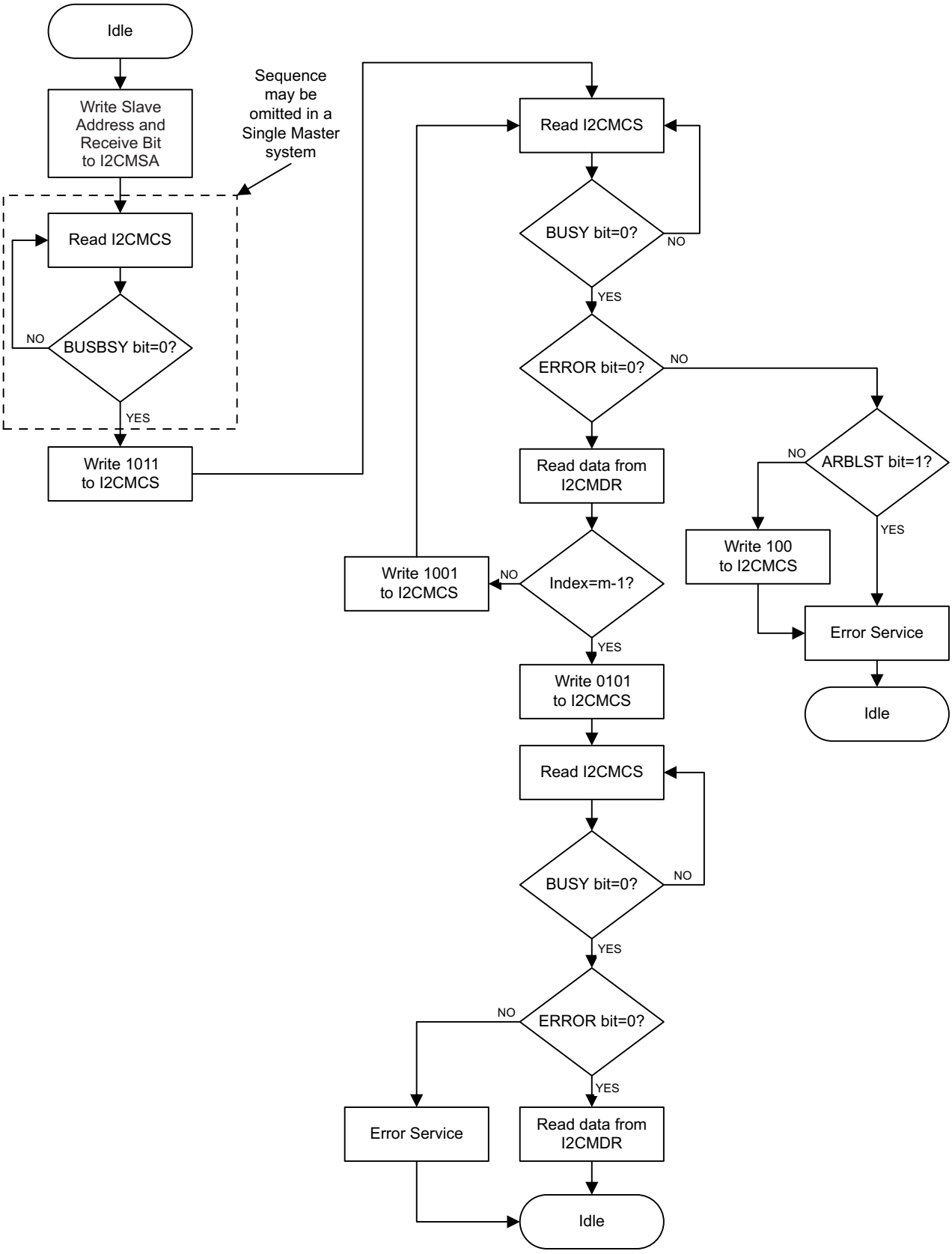
Copyright © 2012–2019, Texas Instruments Incorporated

Figure 22-9. Master TRANSMIT with Repeated START



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 22-10. Master RECEIVE with Repeated START



Copyright © 2012–2019, Texas Instruments Incorporated

Figure 22-11. Master RECEIVE with Repeated START after TRANSMIT with Repeated START

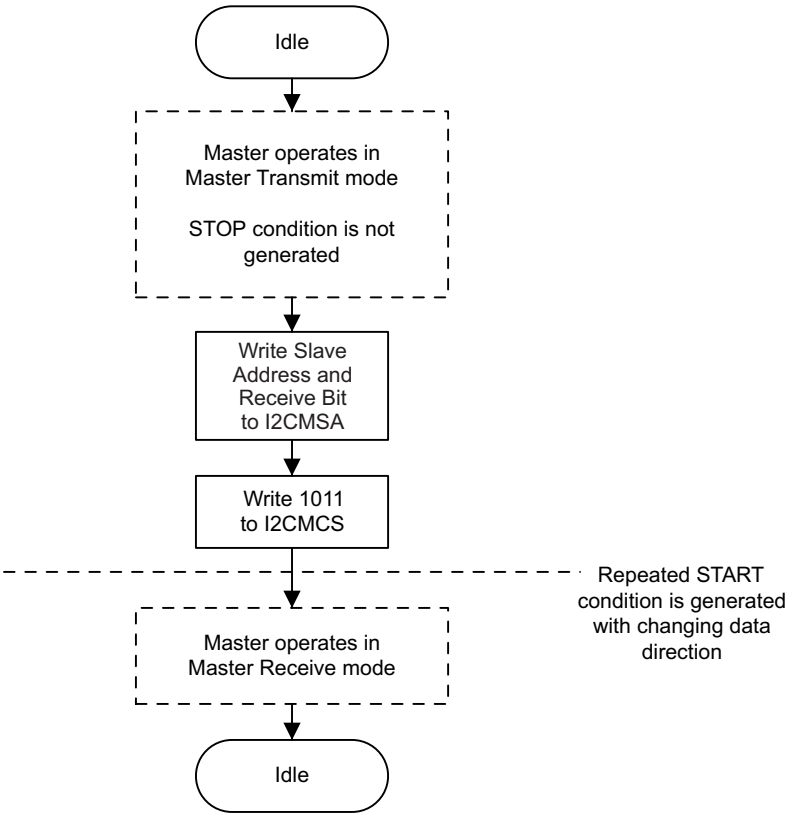
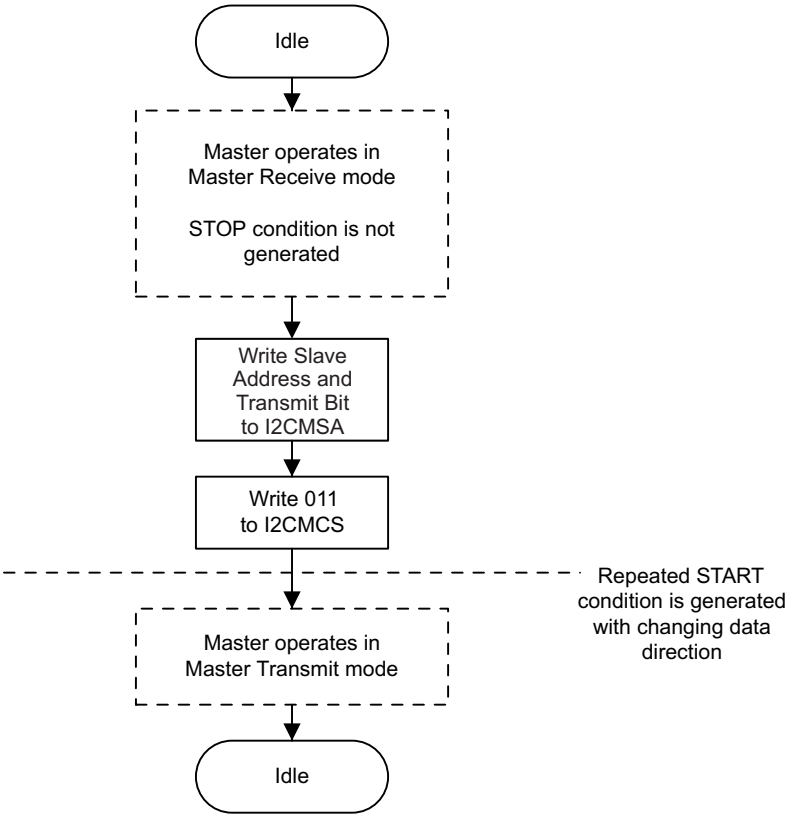


Figure 22-12. Master TRANSMIT with Repeated START after RECEIVE with Repeated START

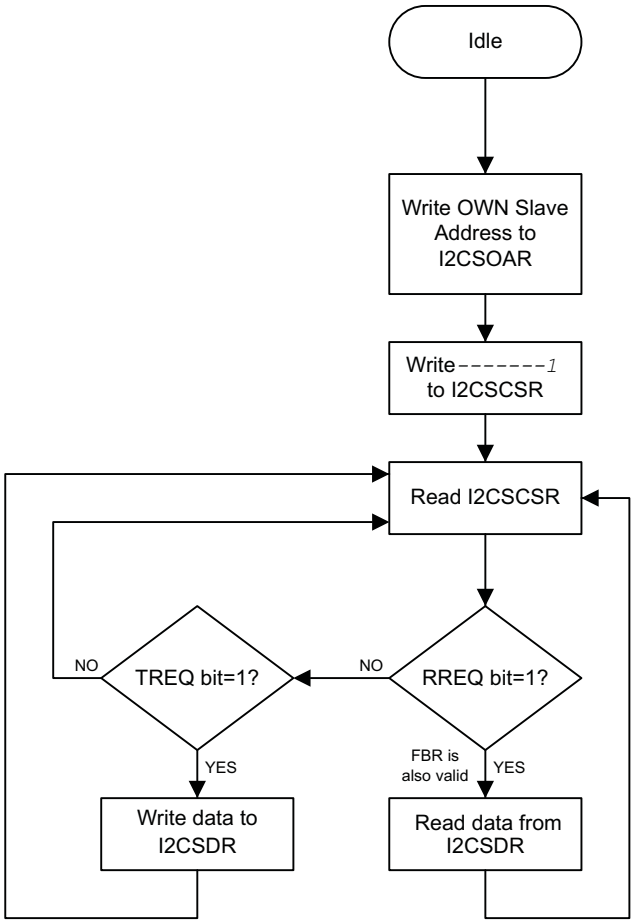


Copyright © 2012–2019, Texas Instruments Incorporated

22.3.5.2 I2C Slave Command Sequences

Figure 22-13 represents the command sequence available for the I2C slave.

Figure 22-13. Slave Command Sequence



Copyright © 2012–2019, Texas Instruments Incorporated

22.4 Initialization and Configuration

The following example shows how to configure the I2C module to transmit a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I2C clock by writing a value of 0x0000.1000 to the RCGC1 register in the *System Control* chapter.
2. Enable the clock to the appropriate GPIO module via the RCGC2 register in the *System Control* chapter. To find out which GPIO port to enable, refer to the *GPIOs* chapter.
3. In the GPIO module, enable the appropriate pins for their alternate function using the GPIOAFSEL register (see the *GPIOs* chapter).
4. Enable the I2C pins for Open Drain operation. See the *GPIOs* chapter.
5. Configure the PMCn fields in the GPIOPCTL register to assign the I2C signals to the appropriate pins. See the *GPIOs* chapter.
6. Initialize the I2C Master by writing the I2CMCR register with a value of 0x0000.0010.
7. Set the desired SCL clock speed of 100 Kbps by writing the I2CMTPR register with the correct value. The value written to the I2CMTPR register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

```
TPR = (System Clock/(2*(SCL_LP + SCL_HP)*SCL_CLK))-1;
TPR = (20MHz/(2*(6+4)*100000))-1;
TPR = 9
```

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.

To generate a single transmit cycle, the I2C Master Slave Address (I2CMSA) register is written with the desired address, the R/S bit is cleared, and the I2C Master Control/Status (I2CMCS) register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due an error), an interrupt becomes active and the data may be read from the I2CMDR register.

When the I2C module operates in Master receiver mode, the ACK bit in the I2CMCS register is normally set, causing the I2C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I2C bus controller requires no further data to be transmitted from the slave transmitter.

Copyright © 2012–2019, Texas Instruments Incorporated

22.5 Register Map

Table 22-2 lists the I2C registers. All addresses given are relative to the I2C base address:

- I2C 0: 0x4002.0000
- I2C 1: 0x4002.1000

Note that the I2C module clock must be enabled before the registers can be programmed (see the *System Control* chapter). There must be a delay of three system clocks after the I2C module clock is enabled before any I2C module registers are accessed.

The hw_i2c.h file in the Concerto™ MWare library uses a base address of 0x800 for the I2C slave registers. Be aware when using registers with offsets between 0x800 and 0x818 that Concerto uses an offset between 0x000 and 0x018 with the slave base address.

Table 22-2. Inter-Integrated Circuit (I2C) Interface Register Map

Offset	Name	Type	Reset	Description
I2C Master				
0x000	I2CMSA	R/W	0x0000.0000	I2C Master Slave Address
0x004	I2CMCS	R/W	0x0000.0000	I2C Master Control/Status
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration
I2C Slave				
0x800	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address
0x804	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status
0x808	I2CSDR	R/W	0x0000.0000	I2C Slave Data
0x80C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask
0x810	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status
0x814	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status
0x818	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear

Copyright © 2012–2019, Texas Instruments Incorporated

22.6 Register Descriptions

The remainder of this section lists and describes the I2S registers, in numerical order by address offset.

22.6.1 I2C Master Slave Address (I2CMSA), offset 0x000

The I2C Master Slave Address (I2CMSA) 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). It is shown and described below.

Figure 22-14. I2C Master Slave Address (I2CMSA) Register

31	16
Reserved	
R-0	

15	8	7	1	0
Reserved		SA		R/S
R-0		R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-3. I2C Master Slave Address (I2CMSA) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-1	SA		I2C Slave Address. Specifies bits A6 through A0 of the slave address
		0	
		1	
0	R/S		Receive/Send. R/S bit specifies if the next operation is a receive (high) or transmit (low).
		0	Transmit
		1	Receive

Copyright © 2012–2019, Texas Instruments Incorporated

22.6.2 I2C Master Control/Status (I2CMCS), offset 0x004

The I2C Master Control/Status (I2CMCS) register accesses status bits when read and control bits when written. When read, the status register indicates the state of the I2C bus controller. When written, the control register configures the I2C controller operation. The first register and description is Read-Only. The second register and description in this section is Write-Only. They are shown and described in the figures and tables below.

Figure 22-15. I2C Master Control/Status (I2CMCS) (Read-Only) Register

31							16
Reserved							
R-0							
15	7	6	5	4	3	2	1	0
Reserved		BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-4. I2C Master Control/Status (I2CMCS) (Read-Only) Register Field Descriptions

Bit	Field	Value	Description
31-7	Reserved		Reserved
6	BUSBSY		Bus busy
		0	The I2C bus is idle.
		1	The I2C bus is busy.
5	IDLE		I2C Idle
		0	The I2C controller is not idle.
		1	The I2C controller is idle.
4	ARBLST		Arbitration Lost
		0	The I2C controller won arbitration.
		1	The I2C controller lost arbitration.
3	DATACK		Acknowledge Data
		0	The transmitted data was acknowledged.
		1	The transmitted data was not acknowledged.
2	ADRACK		Acknowledge Address
		0	The transmitted address was acknowledged.
		1	The transmitted address was not acknowledged.
1	ERROR		Value Description
		0	No error was detected on the last operation.
		1	An error occurred on the last operation.
0	BUSY		I2C Busy
		0	The controller is idle.
		1	The controller is busy.
			Note: When the BUSY bit is set, the other status bits are not valid.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 22-16. I2C Master Control/Status (I2CMCS) (Write-Only) Register

31	4	3	2	1	0
Reserved		ACK	STOP	START	RUN
W-0		W-0	W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-5. I2C Master Control/Status (I2CMCS) Write-Only Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reserved
3	ACK		Data Acknowledge Enable
		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 22-6.
2	STOP		Generate STOP
		0	The controller does not generate the STOP condition.
		1	The controller generates the STOP condition. See field decoding in Table 22-6.
1	START		Generate START
		0	The controller does not generate the START condition.
		1	The controller generates the START or repeated START condition. See field decoding in Table 22-6.
0	RUN		I2C Master Enable
		0	The master is disabled.
		1	The master is enabled to transmit or receive data. See field decoding in Table 22-6.

Table 22-6. 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(1)	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			Illegal
	All other combinations not listed are non-operations.					NOP

(1) An X in a table cell indicates the bit can be 0 or 1.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 22-6. 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 RECEIVE 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

Copyright © 2012–2019, Texas Instruments Incorporated

Table 22-6. Write Field Decoding for I2CMCS[3:0] Field (continued)

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Master Receive	X	0	0	0	1	RECEIVE operation with negative ACK (master remains in Master Receive state).
	X	X	1	0	0	STOP condition (master goes to Idle state).(2)
	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

(2) 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.

Copyright © 2012–2019, Texas Instruments Incorporated

22.6.3 I2C Master Data (I2CMDR), offset 0x008

Important: This register is read-sensitive. See the register description for details.

The I2C Master Data (I2CMDR) register contains the data to be transmitted when in the Master Transmit state and the data received when in the Master Receive state. It is shown and described in the figure and table below.

Figure 22-17. I2C Master Data (I2CMDR) Register

31	8	7	0
Reserved		DATA	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-7. I2C Master Data (I2CMDR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	DATA	00h	Data Transferred Data transferred during transaction.

22.6.4 I2C Master Timer Period (I2CMTPR), offset 0x00C

Th I2C master timer period (I2CMTPR) register specifies the period of the SCL clock. It is shown and described in the figure and table below.

CAUTION

Take care not to set bit 7 when accessing this register as unpredictable behavior can occur.

Figure 22-18. I2C Master Timer Period (I2CMTPR) Register

31	7	6	0
Reserved		TPR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-8. I2C Master Data (I2CMDR) Register Field Descriptions

Bit	Field	Value	Description
31-7	Reserved		Reserved
6-0	TPR	1h	SCL Clock Period SCL_PRD = 2×(1 + TPR)×(SCL_LP + SCL_HP)×CLK_PRD where: *SCL_PRD* is the SCL line period (I2C clock). *TPR* is the Timer Period register value (range of 1 to 127). *SCL_LP* is the SCL Low period (fixed at 6). *SCL_HP* is the SCL High period (fixed at 4). *CLK_PRD* is the system clock period in ns.

Copyright © 2012–2019, Texas Instruments Incorporated

22.6.5 I2C Master Interrupt Mask (I2CMIMR), offset 0x010

The I2C Master Interrupt Mask (I2CMIMR) register controls whether a raw interrupt is promoted to a controller interrupt. It is shown and described in the figure and table below.

Figure 22-19. I2C Master Interrupt Mask (I2CMIMR) Register

31	1	0
Reserved		IM
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-9. I2C Master Interrupt Mask (I2CMIMR) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	IM		Interrupt Mask
		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.

22.6.6 I2C Master Raw Interrupt Status (I2CMRIS), offset 0x014

The I2C Master Raw Interrupt Status (I2CMRIS) register specifies whether an interrupt is pending. It is shown and described in the figure and table below.

Figure 22-20. I2C Master Raw Interrupt Status (I2CMRIS) Register

31	1	0
Reserved		RIS
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-10. I2C Master Raw Interrupt Status (I2CMRIS) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	RIS		Raw Interrupt Status. This bit is cleared by writing a 1 to the IC bit in the I2CMICR register.
		0	No interrupt.
		1	A master interrupt is pending.

Copyright © 2012–2019, Texas Instruments Incorporated

22.6.7 I2C Master Masked Interrupt Status (I2CMMIS), offset 0x018

The I2C Master Masked Interrupt Status (I2CMMIS) register specifies whether an interrupt was signaled. It is shown and described in the figure and table below.

Figure 22-21. I2C Master Masked Interrupt Status (I2CMMIS) Register

31	1	0
Reserved		MIS
R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-11. I2C Master Masked Interrupt Status (I2CMMIS) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	MIS		Masked Interrupt Status. This bit is cleared by writing a 1 to the IC bit in the I2CMICR register.
		0	An interrupt has not occurred or is masked.
		1	An unmasked master interrupt was signaled and is pending.

22.6.8 I2C Master Interrupt Clear (I2CMICR), offset 0x01C

The I2C Master Interrupt Clear (I2CMICR) register clears the raw and masked interrupts. It is shown and described in the figure and table below.

Figure 22-22. I2C Master Interrupt Clear (I2CMICR) Register

31	1	0
Reserved		IC
R-0		W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-12. I2C Master Interrupt Clear (I2CMICR) Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	IC	0	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.

Copyright © 2012–2019, Texas Instruments Incorporated

22.6.9 I2C Master Configuration (I2CMCR), offset 0x020

The I2C Master Configuration (I2CMCR) register configures the mode (master or slave) and sets the interface for test mode loopback. It is shown in the figure and table below.

Figure 22-23. I2C Master Configuration (I2CMCR) Register

31	16
Reserved	
R-0	

15	6	5	4	3	1	0
Reserved		SFE	MFE	Reserved		LPBK
R-0		R/W-0	R/W-0	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-13. I2C Master Configuration (I2CMCR) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5	SFE		I2C Slave Function Enable
		0	Slave mode is disabled.
		1	Slave mode is enabled.
4	MFE		I2C Master Function Enable
		0	Master mode is disabled.
		1	Master mode is enabled.
3-1	Reserved		Reserved
0	LPBK		I2C Loopback
		0	Normal operation
		1	The controller in a test mode loopback configuration.

Copyright © 2012–2019, Texas Instruments Incorporated

22.7 Register Descriptions (I2C Slave)

The remainder of this section lists and describes the I2C slave registers, in numerical order by address offset.

22.7.1 I2C Slave Own Address (I2CSOAR), offset 0x800

The I2C Slave Own Address (I2CSOAR) register consists of seven address bits that identify the I2C device on the I2C bus.

Figure 22-24. I2C Slave Own Address (I2CSOAR) Register

31	7	6	0
Reserved		OAR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-14. I2C Slave Own Address (I2CSOAR) Register Field Descriptions

Bit	Field	Value	Description
31-5			Reserved
6-0		00h	I2C Slave Own Address This field specifies bits A6 through A0 of the slave address.

22.7.2 I2C Slave Control/Status (I2CSCSR), offset 0x804

The I2C Slave Control/Status (I2CSCSR) register functions as a control register when written, and a status register when read.

The first register and description in this section is Read-Only. The second register and description in this section is Write-Only. This register is Read-Only.

Figure 22-25. I2C Slave Control/Status (I2CSCSR) Register (Read-Only)

31	3	2	1	0
Reserved		FBR	TREQ	RREQ
R-0		R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-15. I2C Slave Control/Status (I2CSCSR) Register Field Descriptions (Read-Only)

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	FBR		First Byte Received
		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
1	TREQ		Transmit Request
		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		Receive Request
		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

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 22-26. I2C Slave Control/Status (I2CSCSR) Register (Write-Only)

31	1	0
Reserved		DA
R-0		WO

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-16. I2C Slave Control/Status (I2CSCSR) Register Field Descriptions (Write-Only)

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	DA		Device Active
		0	Disables the I2C slave operation.
		1	Enables the I2C slave operation.

22.7.3 I2C Slave Data (I2CSDR), offset 0x808

Important: This register is read-sensitive. See the register description for details.

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state. It is shown in the figure and table below.

Figure 22-27. I2C Slave Data (I2CSDR) Register

31	8	7	0
Reserved		DATA	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-17. I2C Slave Data (I2CSDR) Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved		Reserved
7-0	DATA	00h	Data for Transfer This field contains the data for transfer during a slave receive or transmit operation.

22.7.4 I2C Slave Interrupt Mask (I2CSIMR), offset 0x80C

The I2C Slave Interrupt Mask (I2CSIMR) register controls whether a raw interrupt is promoted to a controller interrupt. It is shown in the figure and table below.

Figure 22-28. I2C Slave Interrupt Mask (I2CSIMR) Register

31	3	2	1	0
Reserved		STOPIM	STARTIM	DATAIM
R-0		R-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-18. I2C Slave Interrupt Mask (I2CSIMR) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	STOPIM		Stop Condition Interrupt Mask
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 22-18. I2C Slave Interrupt Mask (I2CSIMR) Register Field Descriptions (continued)

Bit	Field	Value	Description
1	STARTIM		Start Condition Interrupt Mask
		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		Data Interrupt Mask
		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

22.7.5 I2C Slave Raw Interrupt Status (I2CSRIS), offset 0x810

The I2C Slave Raw Interrupt Status (I2CSRIS) register specifies whether an interrupt is pending. It is shown in the table and figure below.

Figure 22-29. I2C Slave Raw Interrupt Status (I2CSRIS) Register

31	3	2	1	0
Reserved		STOPRIS	STARTRIS	DATARIS
R-0		R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-19. I2C Slave Raw Interrupt Status (I2CSRIS) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	STOPRIS		Stop Condition Raw Interrupt Status. This bit is cleared by writing a 1 to the STOPIC bit in the I2CSICR register
		0	No interrupt.
		1	A STOP condition interrupt is pending.
1	STARTRIS		Start Condition Raw Interrupt Status. This bit is cleared by writing a 1 to the STARTIC bit in the I2CSICR register.
		0	No interrupt.
		1	A START condition interrupt is pending.
0	DATARIS		Data Raw Interrupt Status. This bit is cleared by writing a 1 to the DATAIC bit in the I2CSICR register.
		0	No interrupt.
		1	A data received or data requested interrupt is pending.

22.7.6 I2C Slave Masked Interrupt Status (I2CSMIS), offset 0x814

The I2C Slave Masked Interrupt Status (I2CSMIS) register specifies whether an interrupt was signaled. It is shown and described in the figure and table below.

Figure 22-30. I2C Slave Masked Interrupt Status (I2CSMIS) Register

31	3	2	1	0
Reserved		STOPMIS	STARTMIS	DATAMIS
R-0		R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 22-20. I2C Slave Masked Interrupt Status (I2CSMIS) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	STOPMIS		Stop Condition Masked Interrupt Status. This bit is cleared by writing a 1 to the STOPIC bit in the I2CSICR register
		0	An interrupt has not occurred or is masked.
		1	An unmasked STOP condition interrupt was signaled is pending.
1	STARTMIS		Start Condition Masked Interrupt Status. This bit is cleared by writing a 1 to the STARTIC bit in the I2CSICR register.
		0	An interrupt has not occurred or is masked.
		1	An unmasked START condition interrupt was signaled is pending
0	DATAMIS		Data Masked Interrupt Status. This bit is cleared by writing a 1 to the DATAIC bit in the I2CSICR register
		0	An interrupt has not occurred or is masked.
		1	An unmasked data received or data requested interrupt was signaled is pending.

22.7.7 I2C Slave Interrupt Clear (I2CSICR), offset 0x818

The I2C Slave Interrupt Clear (I2CSICR) register clears the raw interrupt. A read of this register returns no meaningful data. It is shown in the table and figure below.

Figure 22-31. I2C Slave Interrupt Clear (I2CSICR) Register

31	3	2	1	0
Reserved		STOPIC	STARTIC	DATAIC
R-0		W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-21. I2C Slave Interrupt Clear (I2CSICR) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	STOPIC		Stop Condition Interrupt Clear.
		0	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	0h	Start 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.
0	DATAIC	0h	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.

Copyright © 2012–2019, Texas Instruments Incorporated

M3 Controller Area Network (CAN)

This chapter contains a general description of the Controller Area Network (CAN) module. CAN is a serial communications protocol which efficiently supports distributed real-time control with a high level of reliability. The CAN module supports bit-rates up to 1 Mbit/s and is compliant with the CAN 2.0B protocol specification.

Topic **Page**

23.1 Overview **1513**
23.2 Operating Modes **1515**
23.3 Multiple Clock Source **1520**
23.4 Interrupt Functionality **1520**
23.5 Global Power-down Mode **1521**
23.6 Local Power-down Mode **1521**
23.7 Parity Check Mechanism **1522**
23.8 Debug Mode **1522**
23.9 Module Initialization **1523**
23.10 Configuration of Message Objects **1523**
23.11 Message Handling **1525**
23.12 CAN Bit Timing **1529**
23.13 Message Interface Register Sets **1537**
23.14 Message RAM **1539**
23.15 CAN Control Registers **1543**

Copyright © 2012–2019, Texas Instruments Incorporated

23.1 Overview

This device uses the popular CAN IP known as D_CAN.

23.1.1 Features

The CAN module implements the following features:

- CAN protocol version 2.0 part A, B
- Bit rates up to 1 MBit/s
- Multiple clock sources
- 32 message objects (mailboxes)
- Individual identifier mask for each message object
- Programmable FIFO mode for message objects
- Programmable loop-back modes for self-test operation
- Suspend mode for debug support
- Software module reset
- Automatic bus-on after bus-off state by a programmable 32-bit timer
- Message-RAM parity-check mechanism
- Two interrupt lines
- Globsl power-down and wakeup support

23.1.2 Functional Description

The CAN module performs CAN protocol communication according to ISO 11898-1 (identical to Bosch CAN protocol specification 2.0 A, B). The bit rate can be programmed to values up to 1 MBit/s. A CAN transceiver chip is required for the connection to the physical layer (CAN bus).

For communication on a CAN network, individual message objects can be configured. The message objects and identifier masks are stored in the Message RAM.

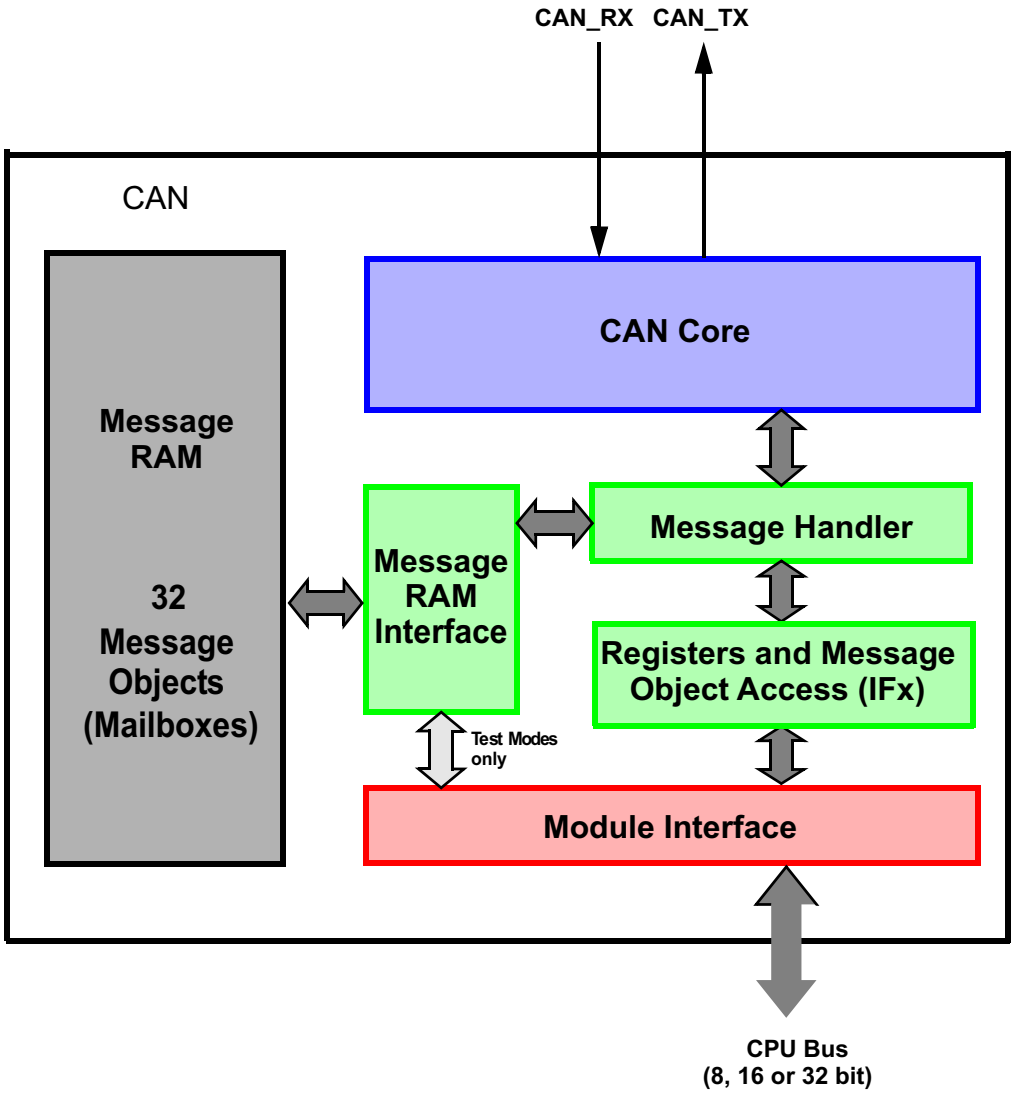
All functions concerning the handling of messages are implemented in the message handler. Those functions are acceptance filtering, the transfer of messages between the CAN Core and the Message RAM, and the handling of transmission requests.

The register set of the CAN can be accessed directly by the CPU via the module interface. These registers are used to control/configure the CAN Core and the message handler, and to access the message RAM.

Copyright © 2012–2019, Texas Instruments Incorporated

23.1.3 Block Diagram

Figure 23-1. CAN Block Diagram



23.1.3.1 CAN Core

The CAN core consists of the CAN Protocol Controller and the Rx/Tx Shift register. It handles all ISO 11898-1 protocol functions.

23.1.3.2 Message Handler

The message handler is a state machine which controls the data transfer between the single ported Message RAM and the CAN Core's Rx/Tx Shift register. It also handles acceptance filtering and the interrupt request generation as programmed in the control registers.

23.1.3.3 Message RAM

The CAN message RAM enables the storage of 32 CAN messages.

23.1.3.4 Registers and Message Object Access (IFx)

Data consistency is ensured by indirect accesses to the message objects. During normal operation, all CPU accesses to the message RAM are done through Interface registers.

Copyright © 2012–2019, Texas Instruments Incorporated

Three interface register sets control the CPU read and write accesses to the Message RAM. There are two interface register sets for read and write access (IF1 and IF2) and one interface register set for read access only (IF3). See also Section 23.13. The interface registers have the same word-length as the message RAM.

In a dedicated test mode, the message RAM is memory-mapped and can be directly accessed.

23.2 Operating Modes

23.2.1 Initialization

The initialization mode is entered either by software (by setting the Init bit in the CAN_CTL register), by hardware reset, or by going bus-off. While the Init bit is set, the message transfer from and to the CAN bus is stopped, and the status of the CAN_TX output is recessive (high). The CAN error counters are not updated. Setting the Init bit does not change any other configuration register.

To initialize the CAN Controller, the CPU has to configure the CAN bit timing and those message objects which have to be used for CAN communication. Message objects which are not needed, can be deactivated by with their MsgVal bits cleared.

The access to the Bit Timing register for the configuration of the bit timing is enabled when both Init and CCE bits in the CAN Control register are set.

Clearing the Init bit finishes the software initialization. Afterwards the 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 (= Bus Idle) before it can take part in bus activities and start the message transfer (for more details see Section 23.12).

The initialization of the message objects is independent of the Init bit; however all message objects should be configured to particular identifiers or set to "not valid" before the message transfer is started.

It is possible to change the configuration of message objects during normal operation by the CPU. After setup and subsequent transfer of the message object from interface registers to the message RAM, the acceptance filtering will be applied to it, when the modified message object number is same or smaller than the previously found message object. This assures data consistency, even when changing message objects, for example, while a pending CAN frame reception.

23.2.2 CAN Message Transfer (Normal Operation)

Once the CAN is initialized and the Init bit is reset to zero, the CAN Core synchronizes itself to the CAN bus and is ready for communication.

Received messages are stored into their appropriate message objects if they pass acceptance filtering. The whole message (MSGID, DLC, and up to eight data bytes) is stored into the message object. As a consequence, when, for example, the identifier mask is used, the MSGID bits which are masked to "don't care" may change in the message object when a received message is stored.

The CPU may read or write each message at any time via the Interface registers, as the message handler guarantees data consistency in case of concurrent accesses.

Messages to be transmitted can be updated by the CPU. If a permanent message object (MSGID and control bits set up during configuration and leaving unchanged for multiple CAN transfers) exists for the message, it is possible to only update the data bytes. If several transmit messages should be assigned to one message object, the whole message object has to be configured before the transmission of this message is requested.

The transmission of multiple message objects may be requested at the same time. They are subsequently transmitted, according to their internal priority. Messages may be updated or set to "not valid" at any time, even if a requested transmission is still pending. However, the data bytes will be discarded if a message is updated before a pending transmission has started.

Depending on the configuration of the message object, a transmission may be automatically requested by the reception of a remote frame with a matching identifier.

Copyright © 2012–2019, Texas Instruments Incorporated

23.2.2.1 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the CAN provides a mechanism to automatically retransmit frames which have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed.

By default, this automatic retransmission is enabled. It can be disabled by setting the DAR bit in the CAN Control register. Further details to this mode are provided in Section 23.11.3.

23.2.2.2 Auto-Bus-On

By default, after the CAN has entered bus-off state, the CPU can start a bus-off-recovery sequence by resetting Init bit. If this is not done, the module will stay in bus-off state.

The CAN provides an automatic auto-bus-on feature which is enabled by the ABO bit in Section 23.15.1. If set, the CAN will automatically start the bus-off-recovery sequence. The sequence can be delayed by a user-defined number of clock cycles which can be defined in Section 23.15.8.

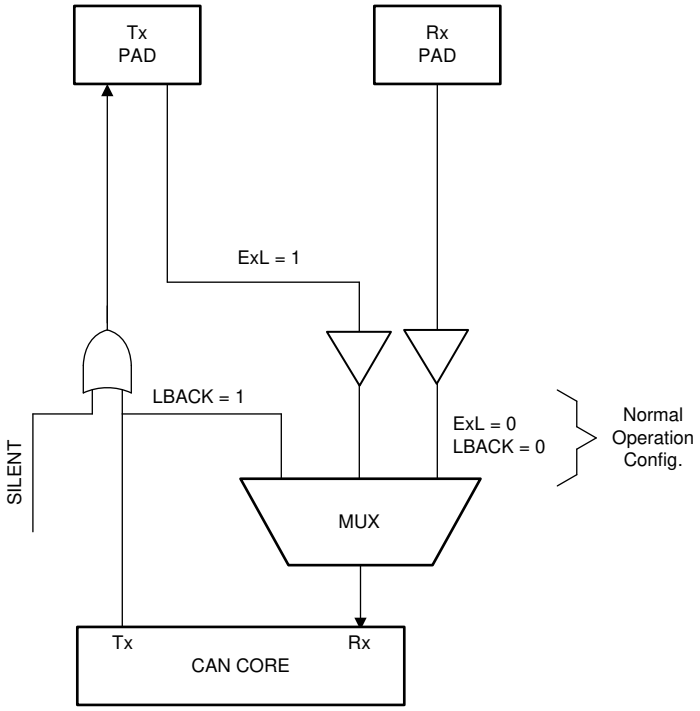
NOTE: If the CAN module goes Bus-Off due to massive occurrence of CAN bus errors, it stops all bus activities and automatically sets the Init bit. Once the Init bit is cleared by the application (or due to the auto-bus-on feature), the device will wait for 128 occurrences of Bus Idle (equal to 128 * 11 consecutive recessive bits) before resuming normal operation. The Bus-Off recovery sequence cannot be shortened by setting or resetting Init bit. At the end of the bus-off recovery sequence, the error counters will be reset. After the Init bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 Error code is written to the Error and Status Register, enabling the CPU to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the proceeding of the Bus-Off recovery sequence.

23.2.3 Test Modes

The CAN module provides several test modes which are mainly intended for self-test purposes. The muxing diagram below aids in understanding the various test modes. This block diagram should be viewed as representative of the module behavior, and not as a gate-accurate implementation of the module. For the sake of brevity, this diagram does not include the GPIO muxing or the I/O buffers.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 23-2. CAN_MUXing



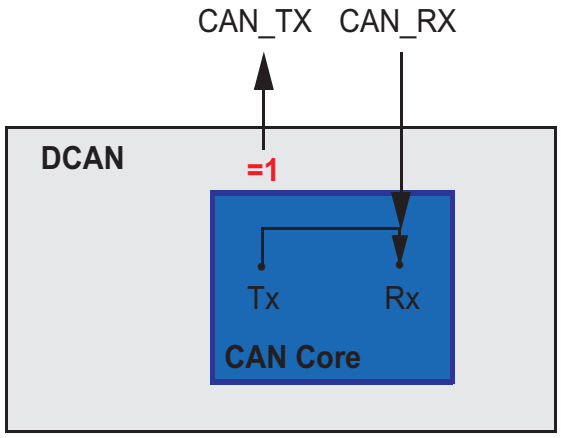
For all test modes, the Test bit in the CAN Control register needs to be set to 1 to enable write access to the Test register (see Section 23.15.6).

23.2.3.1 Silent Mode

The silent mode may be used to analyze the traffic on the CAN bus without affecting it by sending dominant bits (for example, acknowledge bit, overload flag, active error flag). The CAN is still able to receive valid data frames and valid remote frames, but it will not send any dominant bits. However, the received frames are internally routed to the CAN core.

Figure 23-3 shows the connection of signals CAN_TX and CAN_RX to the CAN core in silent mode. Silent mode can be activated by setting the Silent bit in test register (CAN_TEST) to 1. In ISO 11898-1, the silent mode is called the bus monitoring mode.

Figure 23-3. CAN Core in Silent Mode



Copyright © 2012–2019, Texas Instruments Incorporated

23.2.3.2 Loopback Mode

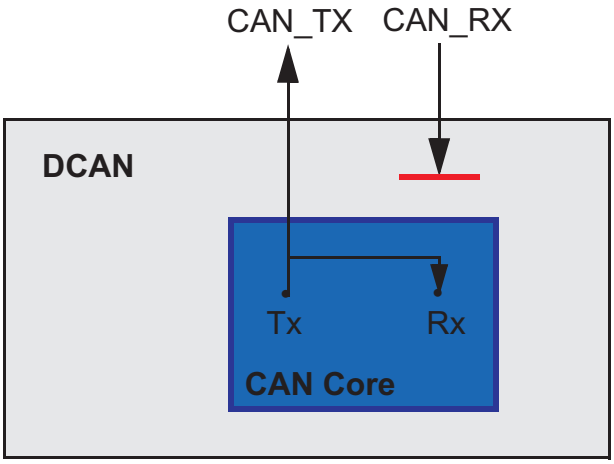
The loopback mode is mainly intended for hardware self-test functions. In this mode, the CAN core uses internal feedback from Tx output to Rx input. Transmitted messages are treated as received messages, and can be stored into message objects if they pass acceptance filtering. The actual value of the CAN_RX input pin is disregarded by the CAN core. Transmitted messages still can be monitored at the CAN_TX pin.

In order to be independent from external stimulation, the CAN core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loopback mode.

Figure 23-4 shows the connection of signals CAN_TX and CAN_RX to the CAN core in loopback mode. Loopback mode can be activated by setting the LBACK bit in the CAN_TEST register to one.

NOTE: In loopback mode, the signal path from CAN core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN Core are disregarded. For including these into the testing, see Section 23.2.3.3.

Figure 23-4. CAN Core in Loopback Mode



23.2.3.3 External Loopback Mode

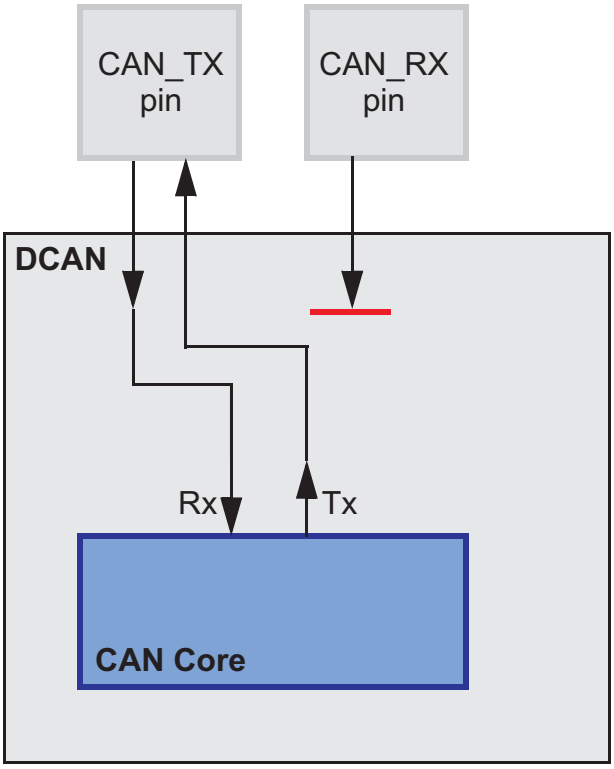
The external loopback mode is similar to the loopback mode; however, it includes the signal path from CAN Core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN Core. When the external loopback mode is selected, the CAN core is connected to the input buffer of the Tx pin. With this configuration, the Tx pin IO circuit can be tested. External loopback mode can be activated by setting the ExL bit in the Test register to one.

Figure 23-5 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in external loopback mode.

NOTE: When loopback mode is active (LBack bit set), the ExL bit will be ignored.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 23-5. CAN Core in External Loopback Mode

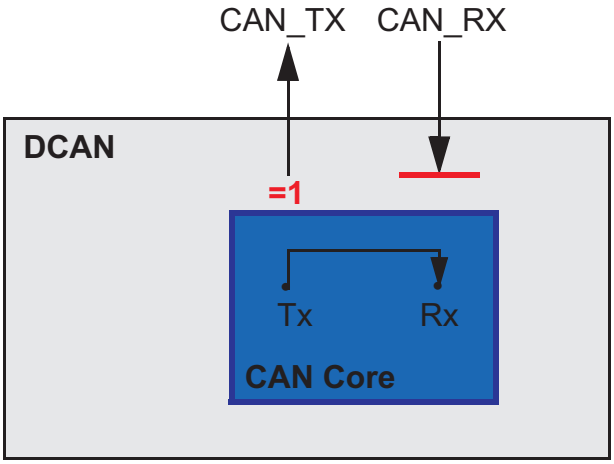


23.2.3.4 Loopback Combined with Silent Mode

It is also possible to combine loopback mode and silent mode by setting the LBack bit and Silent bit at the same time. This mode can be used for a "Hot Selftest", that is, the CAN hardware can be tested without affecting the CAN network. In this mode, the CAN_RX pin is disconnected from the CAN core and no dominant bits will be sent on the CAN_TX pin.

Figure 23-6 shows the connection of the signals CAN_TX and CAN_RX to the CAN Core in case of the combination of loopback mode with silent mode.

Figure 23-6. CAN Core in Loopback Combined with Silent Mode



Copyright © 2012–2019, Texas Instruments Incorporated

23.3 Multiple Clock Source

Three clock domains are provided to the CAN module for generating the CAN bit timing: the external oscillator clock (X1/X2), the M3 system clock, and the GPIO XCLKIN.

The system module reference guide and the device data manual provide for more information on how to configure the relevant clock source registers in the system module.

NOTE: The CAN core has to be programmed to at least eight clock cycles per bit time. To achieve a transfer rate of 1 Mbps an oscillator frequency of 8 MHz or higher has to be used.

23.4 Interrupt Functionality

Interrupts can be generated on two interrupt lines: CAN0INT and CAN1INT. These lines can be enabled by setting IE0 and IE1 bits, respectively, in the CAN Control register.

The CAN provides three groups of interrupt sources: message object interrupts, status change interrupts and error interrupts. The source of an interrupt can be determined by the interrupt identifiers INT0ID and INT1ID in the CAN_INT Interrupt register (see Section 23.15.5). When no interrupt is pending, the register will hold the value zero. Each interrupt line remains active until the dedicated field in the Interrupt register (INT0ID or INT1ID) again reaches zero (this means the cause of the interrupt is reset), or until IE0 or IE1 are reset. The value 0x8000 in the INT0ID field indicates that an interrupt is pending because the CAN core has updated (not necessarily changed) the Error and Status register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The CPU can update (reset) the status bits WakeUpPnd, RxOk, TxOk and LEC by reading the Error and Status register, but a write access of the CPU will never generate or reset an interrupt.

Values between 1 and the number of the last message object indicates that the source of the interrupt is one of the message objects. INT0ID or INT1ID will point to the pending message interrupt with the highest priority. Message Object 1 has the highest priority and the last message object has the lowest priority.

An interrupt service routine which reads the message that is the source of the interrupt, may read the message and reset the message object's IntPnd at the same time (ClrIntPnd bit in the IF1 and IF2 Command register). When IntPnd is cleared, the Interrupt register will point to the next message object with a pending interrupt.

23.4.1 Message Object Interrupts

Message object interrupts are generated by events from the message objects. They are controlled by the flags IntPND, TxIE and RxIE which are described in Section 23.14.1. Message object interrupts can be routed to either CAN0INT or CAN1INT line, controlled by the Interrupt Multiplexer register, see Section 23.15.13. Note that writing to the IntPnd bit in the CAN_IFnMCTL registers can force an interrupt.

23.4.2 Status Change Interrupts

The events WakeUpPnd, RxOk, TxOk and LEC in Error and Status register (see Section 23.15.2) belong to the status change interrupts. The status change interrupt group can be enabled by bit SIE in CAN Control register (see Section 23.15.1). If SIE is set, a status change interrupt will be generated at each CAN frame, independent of bus errors or valid CAN communication, and also independent of the Message RAM configuration. Status Change interrupts can only be routed to interrupt line CAN0INT which has to be enabled by setting IE0 in the CAN Control register.

NOTE: Reading the Error and Status register will clear the WakeUpPnd flag. If in global power-down mode, the WakeUpPnd flag is cleared by such a read access before the CAN module has been woken up by the system, the CAN may reassert the WakeUpPnd flag, and a second interrupt may occur. See also Section 23.5.2).

Copyright © 2012–2019, Texas Instruments Incorporated

23.4.3 Error Interrupts

The events PER, BOff and EWarn (monitored in Section 23.15.2) , belong to the error interrupts. The error interrupt group can be enabled by setting bit EIE in Section 23.15.1). Also,error interrupts can only be routed to interrupt line CAN0INT which has to be enabled by setting IE0 in this register.

23.5 Global Power-down Mode

The Concerto™ architecture supports a centralized global power-down control over the peripheral modules through the SLEEP and DEEP SLEEP modes present in the System Control peripheral.

23.5.1 Entering Global Power-down Mode

The global power down mode for the CAN is requested by configuring the appropriate bits in the sleep mode clock gating control(SCGC) or deep sleep mode clock gating Control(DCGC) register and enabled either of these modes (SLEEP or DEEP SLEEP).

The CAN then finishes all transmit requests of the message objects. When all requests are done, the CAN waits until a bus idle state is recognized. Then it will automatically set the **Init** bit to indicate that the global power-down mode has been entered.

23.5.2 Wakeup from Global Power-down Mode

If the CAN module is in global power-down mode, a CAN bus activity detection circuit is active. On occurrence of a dominant CAN bus level, the CAN will set the **WakeUpPnd** bit in Error and Status register (see Section 23.15.2).

If status interrupts are enabled, also an interrupt will be generated. This interrupt could be used by the application to wakeup the CAN. For this, the application needs to clear the Init bit in CAN Control register.

After the Init bit has been cleared, the CAN module waits until it detects 11 consecutive recessive bits on the CAN_RX pin and then goes bus-active again.

NOTE: The CAN transceiver circuit has to stay active during CAN bus activity detection. The first CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power down mode is lost.

23.6 Local Power-down Mode

Besides from the centralized power-down mechanism controlled by the system control module, the CAN supports a local power-down mode which can be controlled within the DCAN control registers.

23.6.1 Entering Local Power-down Mode

The local power down mode is requested by setting the PDR bit in CAN Control register.

The CAN then finishes all transmit requests of the message objects. When all requests are done, CAN waits until a bus idle state is recognized. Then it will automatically set the Init bit in CAN Control register to prevent any further CAN transfers, and it will also set the PDA bit in CAN Error and Status register. With setting the PDA bits, the CAN module indicates that the local power down mode has been entered.

During local power down mode, the internal clocks of the CAN module are turned off, but there is a wake up logic which can be active, if enabled. Also the actual contents of the control registers can be read back.

23.6.2 Wakeup from Local Power-down Mode

There are two ways to wake up the CAN from local power down mode:

1. The application could wake up the CAN module manually by clearing the PDR bit and then clearing the Init bit in CAN Control register
2. A CAN bus activity detection circuit can be activated by setting the wake up on bus activity bit (WUBA) in CAN Control register.

Copyright © 2012–2019, Texas Instruments Incorporated

If this circuit is active, on occurrence of a dominant CAN bus level, the CAN will automatically start the wake up sequence. It will clear the PDR bit in CAN Control register and also clear the PDA bit in Error and Status register. The WakeUpPnd bit in CAN Error and Status register will be set. If Status Interrupts are enabled, also an interrupt will be generated. Finally the Init bit in CAN control register will be cleared.
After the Init bit has been cleared, the module waits until it detects 11 consecutive recessive bits on the CAN_RX pin and then goes bus-active again.

NOTE: In local low power mode, the application should not clear the Init bit while PDR is set. If there are any messages in the Message RAM configured as to be transmitted and the application resets the init bit, these messages may be sent.In local low power mode, the application should not clear the Init bit while PDR is set. If there are any messages in the Message RAM configured as to be transmitted and the application resets the init bit, these messages may be sent.

23.7 Parity Check Mechanism

The CAN provides a parity check mechanism to ensure data integrity of message RAM data. For each word (32 bits) in Message RAM, one parity bit will be calculated.

Parity information is stored in the Message RAM on write accesses and will be checked against the stored parity bit from Message RAM on read accesses.

The parity check functionality can be enabled or disabled by **PMD** bit field in CAN Control register, see Section 23.15.1. In case of disabled parity check, the parity bits in message RAM will be left unchanged on write access to data area and no check will be done on read access.

If parity checking is enabled, parity bits will be automatically generated and checked by the CAN. A parity bit will be set, if the modulo-2-sum of the data bits is 1. This definition is equivalent to: The parity bit will be set, if the number of 1 bits in the data is odd.

23.7.1 Behavior on Parity Error

On any read access to Message RAM, for example, during start of a CAN frame transmission, the parity of the message object will be checked. If a parity error is detected, the PER bit in Error and Status register will be set. If error interrupts are enabled, an interrupt would also be generated. In order to avoid the transmission of invalid data over the CAN bus, the MsgVal bit of the message object will be reset.

The message object data can be read by the CPU, independently of parity errors. Thus, the application has to ensure that the read data is valid, for example, by immediately checking the Parity Error Code register on parity error interrupt.

23.8 Debug Mode

The module supports the usage of an external debug unit by providing functions like pausing CAN activities and making message RAM content accessible from the debugger. Debug mode is entered automatically when an external debugger is connected and the core is halted.

Before entering Debug mode, the circuit will either wait until a started transmission or reception will be finished and Bus idle state is recognized, or immediately interrupt a current transmission or reception. This is depending on bit **IDS** in Section 23.15.1. Afterwards, the CAN enters Debug mode, indicated by **InitDbg** flag, in the CAN Control register. During debug mode, all CAN registers can be accessed. Reading reserved bits will return '0'. Writing to reserved bits will have no effect. Also, the message RAM will be memory mapped. This allows the external debug unit to read the message RAM. For the memory organization, see Section 23.14.3).

NOTE: During debug mode, the Message RAM cannot be accessed via the IFx register sets.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: Writing to control registers in debug mode may influence the CAN state machine and further message handling.

For debug support, the auto clear functionality of the following CAN registers is disabled:

- Error and Status register (clear of status flags by read)
- IF1 or IF2 Command registers (clear of DMAActive flag by r/w)

23.9 Module Initialization

After hardware reset, the Init bit in the CAN Control register is set and all CAN protocol functions are disabled. The configuration of the bit timing and of the message objects should be completed before the CAN protocol functions are enabled.

For the configuration of the message objects, see Section 23.10.

For the configuration of the Bit Timing, see Section 23.12.2.

The bits MsgVal, NewDat, IntPnd, and TxRqst of the message objects are reset to '0' by a hardware reset. The configuration of a message object is done by programming Mask, Arbitration, Control and Data bits of one of the IF1 or IF2 Interface register sets to the desired values. By writing the message object number to bits [7:0] of the corresponding IF1 or IF2 Command register, the IF1 or IF2 Interface register content is loaded into the addressed message object in the Message RAM.

The configuration of the bit timing requires that the CCE bit in the CAN Control register is set additionally to Init. This is not required for the configuration of the message objects.

When the Init bit in the CAN Control register is cleared, the CAN Protocol Controller state machine of the CAN Core and the message handler State Machine start to control the CAN's internal data flow. Received messages which pass the acceptance filtering are stored into the Message RAM; messages with pending transmission request are loaded into the CAN Core's Shift register and are transmitted via the CAN bus.

The CPU may enable the interrupt lines (setting IE0 and IE1 to '1') at the same time when it clears Init and CCE. The status interrupts EIE and SIE may be enabled simultaneously.

CAN communication may be controlled in interrupt-driven or in polling mode. The Interrupt register points to those message objects with IntPnd = '1'. It is updated even if the interrupt lines to the CPU are disabled (IE0 or IE1 are zero).

The CPU may poll all MessageObject's NewDat and TxRqst bits in parallel from the NewData registers and the Transmission Request registers. Polling can be made easier if all Transmit Objects are grouped at the low numbers, all Receive Objects are grouped at the high numbers.

23.10 Configuration of Message Objects

The whole Message RAM should to be configured before the end of the initialization; however, it is also possible to change the configuration of message objects during CAN communication.

23.10.1 Configuration of a Transmit Object for Data Frames

Figure 23-7 shows how a transmit object can be initialized.

Figure 23-7. Initialization of a Transmit Object

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

- The arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (standard frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored.
- The data registers (DLC[3:0] and Data0-7) are given by the application, TxRqst and RmtEn should not be set before the data is valid.
- If the TxIE bit is set, the IntPnd bit will be set after a successful transmission of the message object.
- If the RmtEn bit is set, a matching received remote frame will cause the TxRqst bit to be set; the

Copyright © 2012–2019, Texas Instruments Incorporated

remote frame will autonomously be answered by a data frame.

- The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask='1') to allow groups of remote frames with similar identifiers to set the TxRqst bit. The Dir bit should not be masked. For details see Section 23.11.8. Identifier masking must be disabled (UMask = '0') if no remote frames are allowed to set the TxRqst bit (RmtEn = '0').

23.10.2 Configuration of a Transmit Object for Remote Frames

It is not necessary to configure transmit objects for the transmission of remote frames. Setting TxRqst for a receive object will cause the transmission of a remote frame with the same identifier as the data frame for which this receive object is configured.

23.10.3 Configuration of a Single Receive Object for Data Frames

Figure 15-7 shows how a receive object for data frames can be initialized.

Figure 23-8. Initialization of a single Receive Object for Data Frames

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

- The arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (Standard Frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a data frame with an 11-bit Identifier is received, ID[17:0] will be set to '0'.
- The data length code (DLC[3:0]) is given by the application. When the message handler stores a data frame in the message object, it will store the received data length code and eight data bytes. If the data length code is less than 8, the remaining bytes of the message object may be overwritten by non specified values.
- The mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask = '1') to allow groups of data frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. If some bits of the Mask bits are set to "don't care", the corresponding bits of the Arbitration register will be overwritten by the bits of the stored data frame.
- If the RxIE bit is set, the IntPnd bit will be set when a received data frame is accepted and stored in the message object.
- If the TxRqst bit is set, the transmission of a remote frame with the same identifier as actually stored in the Arbitration bits will be triggered. The content of the Arbitration bits may change if the Mask bits are used (UMask = '1') for acceptance filtering.

23.10.4 Configuration of a Single Receive Object for Remote Frames

Figure 23-9 shows how a receive object for remote frames can be initialized.

Figure 23-9. Initialization of a single Receive Object for Remote Frames

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	appl.	0	0	0	0

- Receive objects for remote frames may be used to monitor remote frames on the CAN bus. The remote frame stored in the receive object will not trigger the transmission of a data frame. Receive objects for remote frames may be expanded to a FIFO buffer, see Section 23.10.5.
- UMask must be set to '1'. The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be set to "must-match" or to "don't care", to allow groups of remote frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. For details see Section 23.11.8.
- The arbitration bits (ID[28:0] and Xtd bit) may be given by the application. They define the identifier and type of accepted received remote frames. If some bits of the Mask bits are set to "don't care", the corresponding bits of the arbitration bits will be overwritten by the bits of the stored remote frame. If an 11-bit Identifier (standard frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a remote frame with an 11-bit Identifier is received, ID[17:0] will be set to '0'.
- The data length code (DLC[3:0]) may be given by the application. When the message handler stores a

Copyright © 2012–2019, Texas Instruments Incorporated

remote frame in the message object, it will store the received data length code. The data bytes of the message object will remain unchanged.

- If the RxIE bit is set, the IntPnd bit will be set when a received remote frame is accepted and stored in the message object.

23.10.5 Configuration of a FIFO Buffer

With the exception of the EoB bit, the configuration of receive objects belonging to a FIFO buffer is the same as the configuration of a single receive object.

To concatenate multiple message objects to 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 number will be the first message object of the FIFO buffer. The EoB bit of all message objects of a FIFO buffer except the last one have to be programmed to zero. The EoB bits of the last message object of a FIFO buffer is set to one, configuring it as the end of the block.

23.11 Message Handling

When initialization is finished, the CAN module synchronizes itself to the traffic on the CAN bus. It does acceptance filtering on received messages and stores those frames that are accepted into the designated message objects. The application has to update the data of the messages to be transmitted and to enable and request their transmission. The transmission is requested automatically when a matching remote frame is received.

The application may read messages which are received and accepted. Messages that are not read before the next messages is accepted for the same message object will be overwritten. Messages may be read interrupt-driven or after polling of NewDat.

23.11.1 Message Handler Overview

The message handler state machine controls the data transfer between the Rx/Tx Shift register of the CAN Core and the Message RAM. It performs the following tasks:

- Data transfer from Message RAM to CAN Core (messages to be transmitted).
- Data transfer from CAN Core to the Message RAM (received messages).
- Data transfer from CAN Core to the Acceptance Filtering unit.
- Scanning of Message RAM for a matching message object (acceptance filtering).
- Scanning the same message object after being changed by IF1 or IF2 registers when priority is same or higher as message the object found by last scanning.
- Handling of TxRqst flags.
- Handling of interrupt flags.

The message handler registers contains status flags of all message objects grouped into the following topics:

- Transmission request flags
- New data flags
- Interrupt pending flags
- Message valid registers

Instead of collecting above listed status information of each message object via IFx registers separately, these message handler registers provides a fast and easy way to get an overview e.g. about all pending transmission requests.

All message handler registers are read-only.

Copyright © 2012–2019, Texas Instruments Incorporated

23.11.2 Receive/Transmit Priority

The receive/transmit priority for the message objects is attached to the message number, not to the CAN identifier. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding message object, so for example, messages with the highest priority can be placed in the message objects with the lowest numbers.

The acceptance filtering for received data frames or remote frames is also done in ascending order of message objects, so a frame that has been accepted by a message object cannot be accepted by another message object with a higher message number. The last message object may be configured to accept any data frame or remote frame that was not accepted by any other message object, for nodes that need to log the complete message traffic on the CAN bus.

23.11.3 Transmission of Messages in Event Driven CAN Communication

If the shift register of the CAN Core is ready for loading and if there is no data transfer between the IFx registers and Message RAM, the MsgVal bits in the Message Valid register and the TxRqst bits in the transmission request register are evaluated. The valid message object with the highest priority pending transmission request is loaded into the shift register by the message handler and the transmission is started. The message object's NewDat bit is reset.

After a successful transmission and if no new data was written to the message object (NewDat = '0') since the start of the transmission, the TxRqst bit will be reset. If TxIE is set, IntPnd will be set after a successful transmission. If the CAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

If automatic retransmission mode is disabled by setting the DAR bit in the CAN Control register, the behavior of bits TxRqst and NewDat in the Message Control register of the Interface register set is as follows:

- When a transmission starts, the TxRqst bit of the respective Interface register set is reset, while bit NewDat remains set.
- When the transmission has been successfully completed, the NewDat bit is reset.

When a transmission failed (lost arbitration or error) bit NewDat remains set. To restart the transmission, the application has to set TxRqst again.

Received remote frames do not require a receive object. They will automatically trigger the transmission of a data frame, if in the matching Transmit Object the RmtEn bit is set.

23.11.4 Updating a Transmit Object

The CPU may update the data bytes of a transmit object any time via the IF1 or IF2 interface registers, neither MsgVal nor TxRqst have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes in the corresponding IF1 or IF2 Data A register, or the IF1 or IF2 Data B register have to be valid before the content of that register is transferred to the message object. Either the CPU has to write all four bytes into the IF1 or IF2 Data register, or the message object is transferred to the IF1 or IF2 Data register before the CPU writes the new data bytes.

When only the data bytes are updated, first 0x87 can be written to bits [23:16] of the Command register and then the number of the message object is written to bits [7:0] of the Command register, concurrently updating the data bytes and setting TxRqst with NewDat.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat has to be set together with TxRqst in event driven CAN communication. For details see Section 23.11.3.

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

Copyright © 2012–2019, Texas Instruments Incorporated

23.11.5 Changing a Transmit Object

If the number of implemented message objects is not sufficient to be used as permanent message objects only, the transmit objects may be managed dynamically. The CPU can write the whole message (Arbitration,

Control, and Data) into the Interface register. The bits [23:16] of the Command register can be set to 0xB7 for the transfer of the whole message object content into the message object. Neither MsgVal nor TxRqst have to be reset before this operation.

If a previously requested transmission of this message object is not completed but already in progress, it will be continued; however it will not be repeated if it is disturbed.

To only update the data bytes of a message to be transmitted, bits [23:16] of the Command register should be set to 0x87.

NOTE: After the update of the transmit object, the interface register set will contain a copy of the actual contents of the object, including the part that had not been updated.

23.11.6 Acceptance Filtering of Received Messages

When the arbitration and control bits (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the shift register of the CAN Core, the message handler starts to scan of the message RAM for a matching valid message object:

- The acceptance filtering unit is loaded with the arbitration bits from the CAN Core shift register.
- Then the arbitration and mask bits (including MsgVal, UMask, NewDat, and EoB) of Message Object 1 are loaded into the Acceptance Filtering unit and are compared with the arbitration bits from the shift register. This is repeated for all following message objects until a matching message object is found, or until the end of the Message RAM is reached.
- If a match occurs, the scanning is stopped and the message handler proceeds depending on the type of the frame (data frame or remote frame) received.

23.11.7 Reception of Data Frames

The message handler stores the message from the CAN Core shift register into the respective message object in the Message RAM. Not only the data bytes, but all arbitration bits and the data length code are stored into the corresponding message object. This ensures that the data bytes stay associated to the identifier even if arbitration mask registers are used.

The NewDat bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset the NewDat bit when it reads the message object. If at the time of the reception the NewDat bit was already set, MsgLst is set to indicate that the previous data (supposedly not seen by the CPU) is lost. If the RxIE bit is set, the IntPnd bit is set, causing the Interrupt register to point to this message object.

The TxRqst bit of this message object is reset to prevent the transmission of a remote frame, while the requested data frame has just been received.

23.11.8 Reception of Remote Frames

When a remote frame is received, three different configurations of the matching message object have to be considered:

1. Dir = '1' (direction = transmit), RmtEn = '1', UMask = '1' or '0'

   The TxRqst bit of this message object is set at the reception of a matching remote frame. The rest of the message object remains unchanged.
2. Dir = '1' (direction = transmit), RmtEn = '0', UMask = '0'

   The remote frame is ignored, this message object remains unchanged.
3. Dir = '1' (direction = transmit), RmtEn = '0', UMask = '1'

   The remote frame is treated similar to a received data frame. At the reception of a matching remote

Copyright © 2012–2019, Texas Instruments Incorporated

frame, the TxRqst bit of this message object is reset. The arbitration and control bits (Identifier + IDE + RTR + DLC) from the shift register are stored in the message object in the Message RAM and the NewDat bit of this message object is set. The data bytes of the message object remain unchanged

23.11.9 Reading Received Messages

The CPU may read a received message any time via the IFx interface registers, the data consistency is guaranteed by the message handler state machine.

Typically the CPU will write first 0x7F to bits [23:16] and then the number of the message object to bits [7:0] of the Command register. That combination will transfer the whole received message from the Message RAM into the Interface register set. Additionally, the bits NewDat and IntPnd are cleared in the Message RAM (not in the Interface register set). The values of these bits in the Message Control register always reflect the status before resetting the bits.

If the message object uses masks for acceptance filtering, the arbitration bits show which of the different matching messages has been received.

The actual value of NewDat shows whether a new message has been received since last time when this message object was read. The actual value of MsgLst shows whether more than one message have been received since the last time when this message object was read. MsgLst will not be automatically reset.

23.11.10 Requesting New Data for a Receive Object

By means of a remote frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TxRqst bit of a receive object will cause 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.

Setting the TxRqst bit without changing the contents of a message object requires the value 0x84 in bits [23:16] of the Command register.

23.11.11 Storing Received Messages in FIFO Buffers

Several message objects may be grouped to form one or more FIFO Buffers. Each FIFO Buffer configured to store received messages with a particular (group of) Identifier(s). Arbitration and Mask registers of the FIFO Buffer's message objects are identical. The EoB (End of Buffer) bits of all but the last of the FIFO Buffer's message objects are '0', in the last one the EoB bit is '1'.

Received messages with identifiers matching to a FIFO Buffer are stored into a message object of this FIFO Buffer, 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 bit of this message object is set. By setting NewDat while EoB is '0' the message object is locked for further write accesses by the message handler until the CPU has cleared the NewDat bit.

Messages are stored into a FIFO Buffer until the last message object of this FIFO Buffer is reached. If none of the preceding message objects is released by writing NewDat to '0', all further messages for this FIFO Buffer will be written into the last message object of the FIFO Buffer (EoB = '1') and therefore overwrite previous messages in this message object.

23.11.12 Reading from a FIFO Buffer

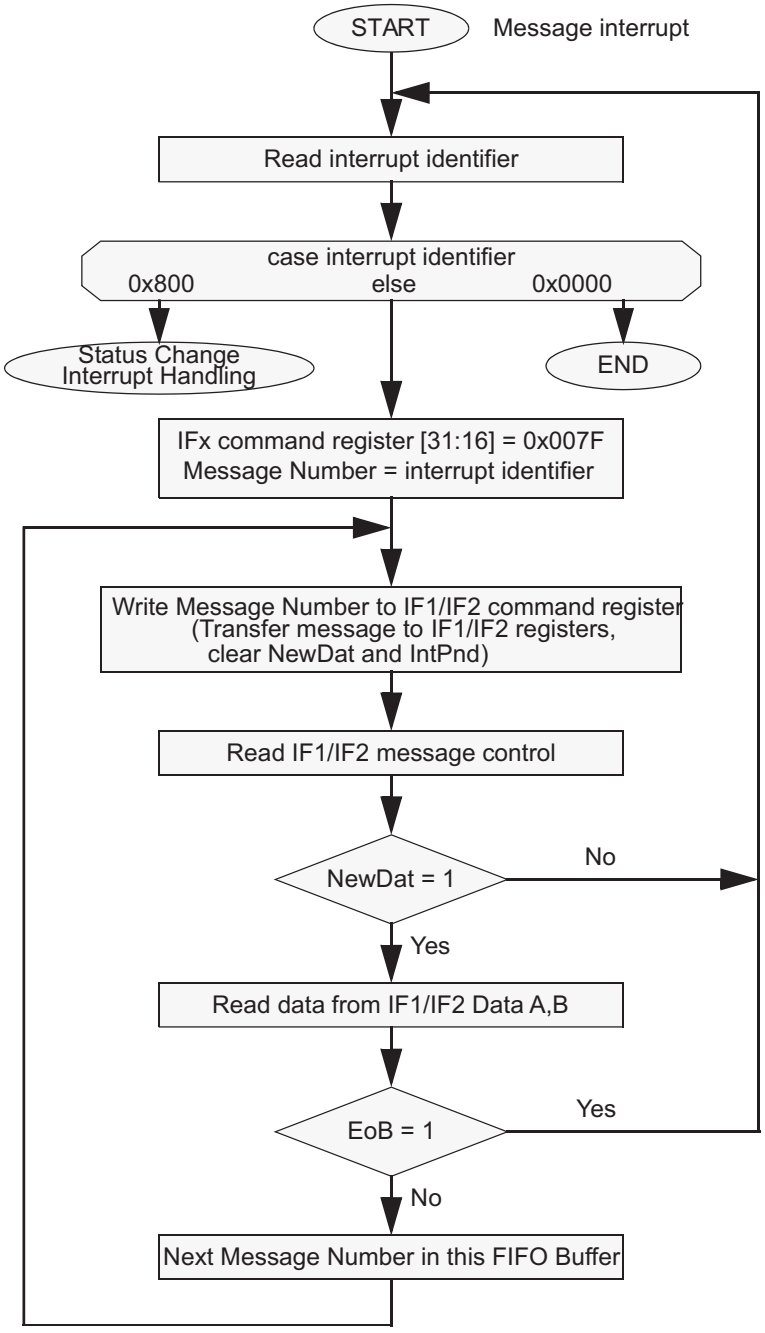
Several messages may be accumulated in a set of message objects which are concatenated to form a FIFO Buffer before the application program is required (in order to avoid the loss of data) to empty the buffer. A FIFO Buffer of length N will store N-1 plus the last received message since last time it was cleared. A FIFO Buffer is cleared by reading and resetting the NewDat bits of all its message objects, starting at the FIFO Object with the lowest message number. This should be done in a subroutine following the example shown in Figure 23-10.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: All message objects of a FIFO buffer needs to be read and cleared before the next batch of messages can be stored. Otherwise true FIFO functionality cannot be guaranteed, since the message objects of a partly read buffer will be re-filled according to the normal (descending) priority.

Reading from a FIFO Buffer message object and resetting its NewDat bit is handled the same way as reading from a single message object.

Figure 23-10. CPU Handling of a FIFO Buffer (Interrupt Driven)



23.12 CAN Bit Timing

The CAN supports bit rates between 1 kBit/s and 1000 kBit/s.

Copyright © 2012–2019, Texas Instruments Incorporated

Each member of the CAN network has its own clock generator, typically derived from a crystal oscillator. The Bit timing parameters can be configured individually for each CAN node, creating a common Bit rate even though the CAN nodes' oscillator periods (fosc) may be different.

The frequencies of these oscillators are not absolutely stable. Small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range (df), the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream.

In many cases, the CAN bit synchronization will amend 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.

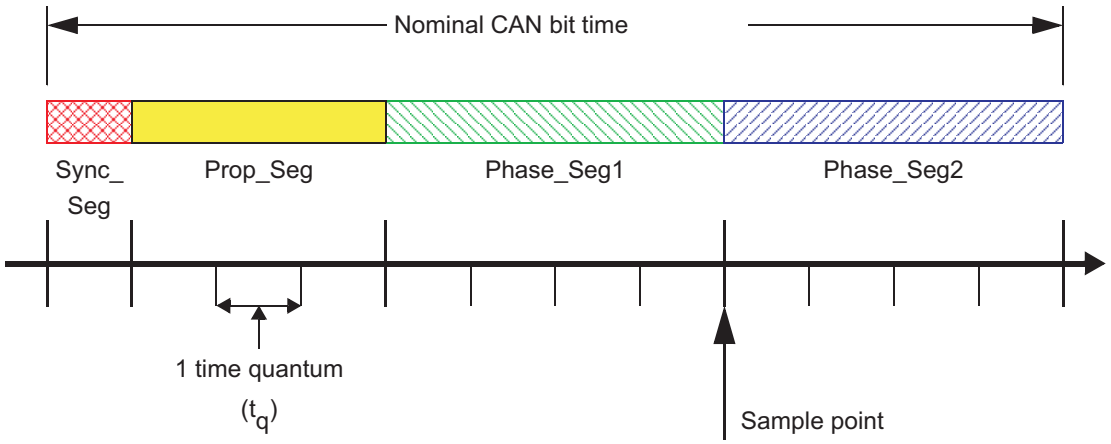
Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

23.12.1 Bit Time and Bit Rate

According to the CAN specification, the Bit time is divided into four segments (see Figure 23-11Figure 15-10):

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

Figure 23-11. Bit Timing



Each segment consists of a specific number of time quanta. The length of one time quantum (t_q), which is the basic time unit of the bit time, is given by the CAN_CLK and the Baud Rate Prescalers (BRPE and BRP). With these two Baud Rate Prescalers combined, divider values from 1 to 1024 can be programmed:

t_q = Baud Rate Prescaler / CAN_CLK

Apart from the fixed length of the synchronization segment, these numbers are programmable. Table 23-1 describes the minimum programmable ranges required by the CAN protocol.

A given bit rate may be met by different bit time configurations.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-1. Programmable Ranges Required by CAN Protocol

Parameter	Range	Remark
Sync_Seg	1 t_q (fixed)	Synchronization of bus input to CAN_CLK
Prop_Seg	[1 … 8] t_q	Compensates for the physical delay times
Phase_Seg1	[1 … 8] t_q	May be lengthened temporarily by synchronization
Phase_Seg2	[1 … 8] t_q	May be shortened temporarily by synchronization
Synchronization Jump Width (SJW)	[1 … 4] t_q	May not be longer than either Phase Buffer Segment

NOTE: For proper functionality of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

23.12.1.1 Synchronization Segment

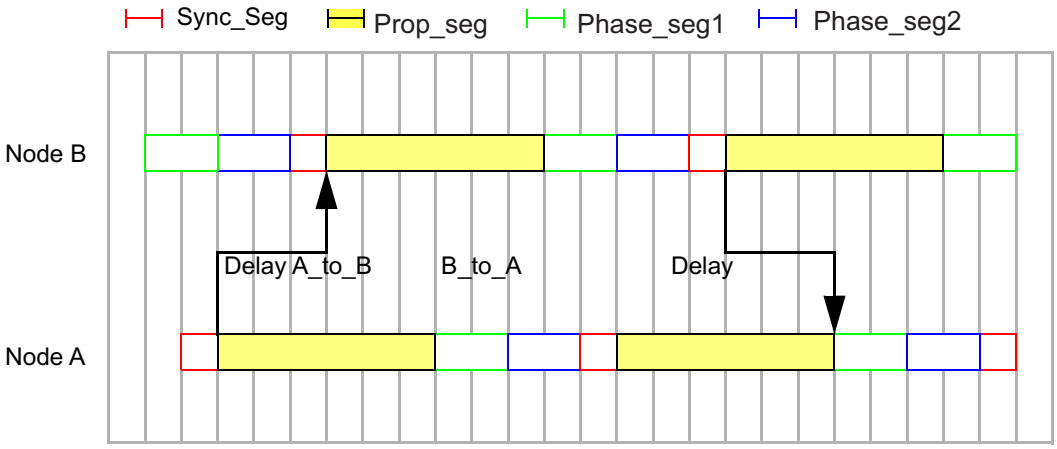
The Synchronization Segment (Sync_Seg) is the part of the bit time where edges of the CAN bus level are expected to occur. If an edge occurs outside of Sync_Seg, its distance to the Sync_Seg is called the phase error of this edge.

23.12.1.2 Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the CAN network. These delay times consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus can be out of phase with the transmitter of the bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's nondestructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages require that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in Figure 23-12 shows the phase shift and propagation times between two CAN nodes.

Figure 23-12. The Propagation Time Segment



Delay A_to_B >= node output delay(A) + bus line delay(A→B) + node input delay(B)

Prop_Seg >= Delay A_to_B + Delay B_to_A

Prop_Seg >= 2 • [max(node output delay+ bus line delay + node input delay)]

Copyright © 2012–2019, Texas Instruments Incorporated

In this example, both nodes A and B are transmitters performing an arbitration for the CAN bus. The node A has sent its Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay(A_to_B) after it has been transmitted, node B's bit timing segments are shifted with regard to node A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay(B_to_A).

Due to oscillator tolerances, the actual position of node A's Sample Point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase_Seg1. This condition defines the length of Prop_Seg.

If the edge from recessive to dominant transmitted by node B would arrive at node A after the start of Phase_Seg1, it could happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

This error only occurs when two nodes arbitrate for the CAN bus which have oscillators of opposite ends of the tolerance range and are separated by a long bus line; this is an example of a minor error in the Bit timing configuration (Prop_Seg too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode. The CAN does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of 1 t_q, requiring a longer Prop_Seg.

23.12.1.3 Phase Buffer Segments and Synchronization

The phase buffer segments (Phase_Seg1 and Phase_Seg2) and the synchronization jump width (SJW) are used to compensate for the oscillator tolerance.

The phase buffer segments surround the sample point. The phase buffer segments may be lengthened or shortened by synchronization.

The synchronization jump width (SJW) defines how far the resynchronizing mechanism may move the sample point inside the limits defined by the phase buffer segments to compensate for edge phase errors.

Synchronizations occur on edges from recessive to dominant. Their purpose is to control the distance between edges and sample points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous sample point. A synchronization may be done only if a recessive bit was sampled at the previous sample point and if the actual time quantum's bus level is dominant.

An edge is synchronous if it occurs inside of Sync_Seg, otherwise its distance to the Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist: hard synchronization and resynchronizing. A hard synchronization is done once at the start of a frame; inside a frame only resynchronization is possible.

- Hard Synchronization

  After a hard synchronization, the bit time is restarted with the end of Sync_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time.

- Bit Resynchronizations

  Resynchronization leads to a shortening or lengthening of the bit time such that the position of the sample point is shifted with regard to the edge.

  When the phase error of the edge which causes resynchronization is positive, Phase_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.

  When the phase error of the edge which causes Resynchronization is negative, Phase_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

Copyright © 2012–2019, Texas Instruments Incorporated

If the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of hard synchronization and resynchronization are the same. If the magnitude of the phase error is larger than SJW, the resynchronization cannot compensate the phase error completely, and an error of (phase error - SJW) remains.

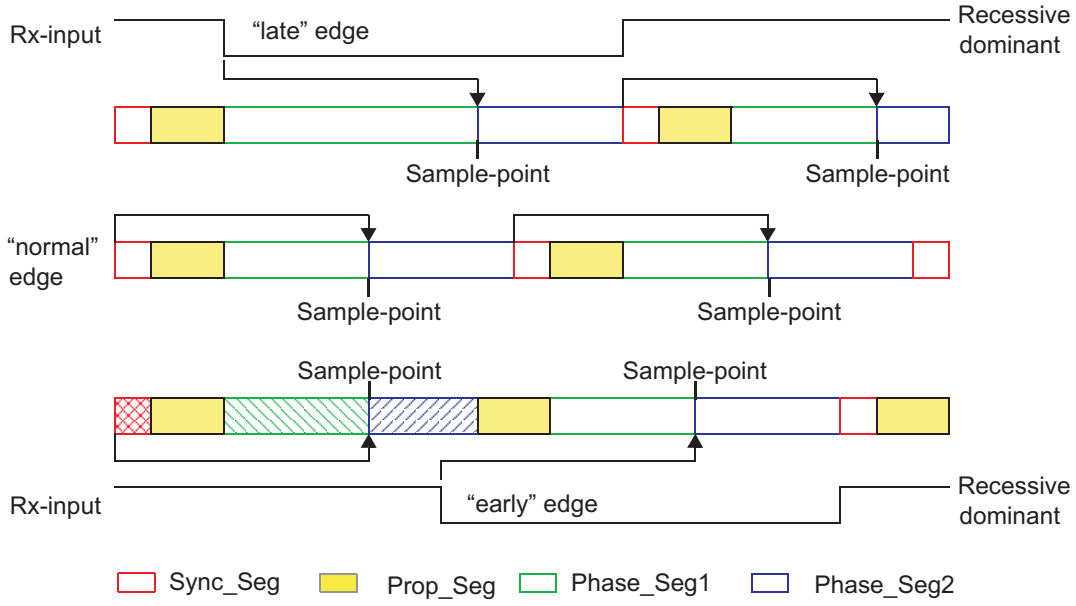
Only one synchronization may be done between two sample points. The synchronizations maintain a minimum distance between edges and sample points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop_Seg + Phase_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize "hard" on the edge transmitted by the "leading" transceiver that started transmitting first, but due to propagation delay times, they cannot become ideally synchronized. The "leading" transmitter does not necessarily win the arbitration, therefore the receivers have to synchronize themselves to different transmitters that subsequently "take the lead" and that are differently synchronized to the previously "leading" transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that "takes the lead" in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator's clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator's tolerance range.

The examples in Figure 23-13 show how the phase buffer segments are used to compensate for phase errors. There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a "late" edge, the lower drawing shows the synchronization on an "early" edge, and the middle drawing is the reference without synchronization.

Figure 23-13. Synchronization on Late and Early Edges



In the first example, an edge from recessive to dominant occurs at the end of Prop_Seg. The edge is "late" since it occurs after the Sync_Seg. Reacting to the "late" edge, Phase_Seg1 is lengthened so that the distance from the edge to the sample point is the same as it would have been from the Sync_Seg to the sample point if no edge had occurred. The phase error of this "late" edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync_Seg.

In the second example, an edge from recessive to dominant occurs during Phase_Seg2. The edge is "early" since it occurs before a Sync_Seg. Reacting to the "early" edge, Phase_Seg2 is shortened and Sync_Seg is omitted, so that the distance from the edge to the sample point is the same as it would have been from a Sync_Seg to the sample point if no edge had occurred. As in the previous example, the magnitude of this "early" edge's phase error is less than SJW, so it is fully compensated.

Copyright © 2012–2019, Texas Instruments Incorporated

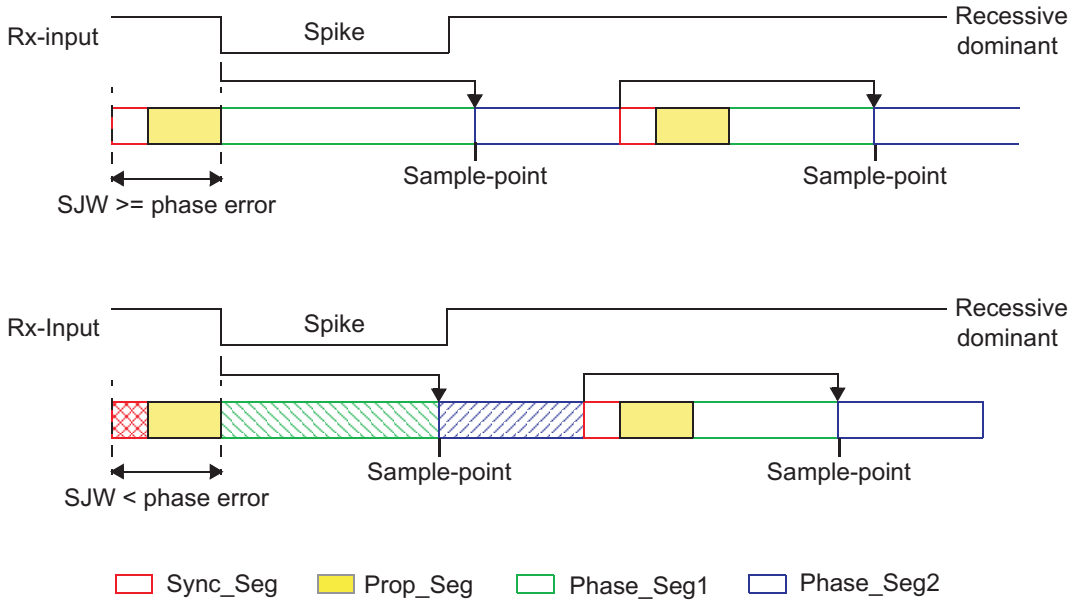
The phase buffer segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

In these examples, the bit timing is seen from the point of view of the CAN implementation's state machine, where the bit time starts and ends at the sample points. The state machine omits Sync_Seg when synchronizing on an "*early*" edge because it cannot subsequently redefine that time quantum of Phase_Seg2 where the edge occurs to be the Sync_Seg.

The examples in Figure 23-14 show how short dominant noise spikes are filtered by synchronizations. In both examples, the spike starts at the end of Prop_Seg and has the length of (Prop_Seg + Phase_Seg1). In the first example, the synchronization jump width is greater than or equal to the phase error of the spike's edge from recessive to dominant. Therefore the sample point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the sample point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

Figure 23-14. Filtering of Short Dominant Spikes



23.12.1.4 Oscillator Tolerance Range

With the introduction of CAN protocol version 1.2, the option to synchronize on edges from dominant to recessive became obsolete. Only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range df for an oscillator's frequency f_{osc} around the nominal frequency f_{nom} with

$(1 - df) \bullet f_{nom} \leq f_{osc} \leq (1 + df) \bullet f_{nom}$

depends on the proportions of Phase_Seg1, Phase_Seg2, SJW, and the bit time. The maximum tolerance df is the defined by two conditions (both shall be met):

(11)

$$\text{I:} \quad df \leq \frac{\min(TSeg1, Tseg2)}{2 \times (13 \times (bit\_time - TSeg2))}$$

(12)

$$\text{II:} \quad df \leq \frac{SJW}{20 \times bit\_time}$$

Copyright © 2012–2019, Texas Instruments Incorporated

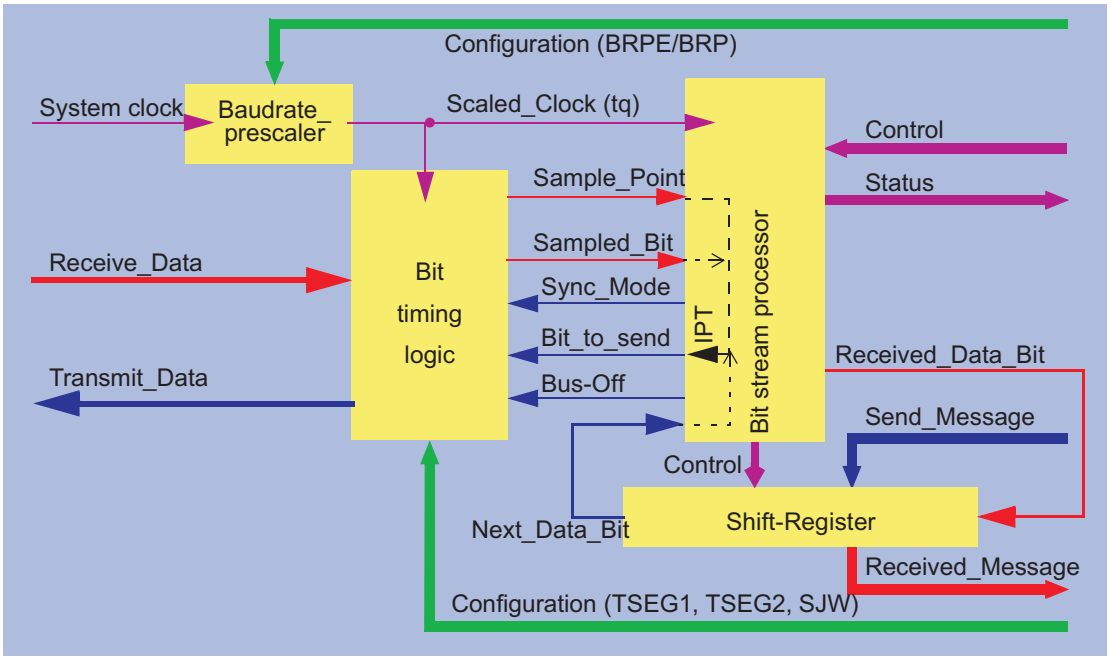
It has to be considered that SJW may not be larger than the smaller of the phase buffer segments and that the propagation time segment limits that part of the bit time that may be used for the phase buffer segments.

The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kBit/s (bit time = 8 µs) with a bus length of 40 m.

23.12.2 Configuration of the CAN Bit Timing

In the CAN, the bit timing configuration is programmed in two register bytes, additionally a third byte for a baud rate prescaler extension of 4 bits (BRPE) is provided. The sum of Prop_Seg and Phase_Seg1 (as TSEG1) is combined with Phase_Seg2 (as TSEG2) in one byte, SJW and BRP (plus BRPE in third byte) are combined in the other byte (see Figure 23-15).

Figure 23-15. Structure of the CAN Core's CAN Protocol Controller



In this bit timing register, the components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1…n], values in the range of [0…n-1] are programmed. That way, e.g. SJW (functional range of [1…4]) is represented by only two bits.

Therefore the length of the Bit time is (programmed values) [TSEG1 + TSEG2 + 3] t_q or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

The data in the Bit Timing register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by BRPE/BRP) 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 Bit timing state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the Bit Stream Processor (BSP) state machine, is evaluated once each bit time, at the Sample Point.

The Shift register serializes the messages to be sent and parallelizes received messages. Its loading and shifting is controlled by the BSP.

Copyright © 2012–2019, Texas Instruments Incorporated

The BSP translates messages into frames and vice versa. It 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. It is evaluated at the sample point and processes the sampled bus input bit. The time after the sample point that is needed to calculate the next bit to be sent (e.g. data bit, CRC bit, stuff bit, error flag, or idle) is called the Information Processing Time (IPT), which is 0 t_q for the CAN.

Generally, the IPT is CAN controller specific, but may not be longer than 2 t_q. The IPC length is the lower limit of the programmed length of Phase_Seg2. In case of a synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

23.12.2.1 Calculation of the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting Bit time (1 / Bit rate) must be an integer multiple of the CAN clock period.

NOTE: 8 MHz is the minimum CAN clock frequency required to operate the CAN at a bit rate of 1 MBit/s.

The bit time may consist of 8 to 25 time quanta. The length of the time quantum t_q is defined by the Baud Rate Prescaler with t_q = (Baud Rate Prescaler) / CAN_CLK. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandible CAN bus systems. The resulting time for Prop_Seg is converted into time quanta (rounded up to the nearest integer multiple of t_q).

The Sync_Seg is 1 t_q long (fixed), leaving (bit time - Prop_Seg - 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 may not be shorter than any CAN controller's Information Processing Time in the network, which is device dependent and can be in the range of [0…2] t_q.

The length of the synchronization jump width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in Section 23.12.1.4.

If more than one configurations are possible to reach a certain Bit rate, it is recommended to choose the configuration which allows the highest oscillator tolerance range.

CAN nodes with different 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.

The resulting configuration is written into the Bit Timing register:

(Phase_Seg2-1)&(Phase_Seg1+Prop_Seg-1)&

(SynchronizationJumpWidth-1)&(Prescaler-1)

23.12.2.2 Example for Bit Timing at high Baudrate

In this example, the frequency of CAN_CLK is 10 MHz, BRP is 0, the bit rate is 1 MBit/s.

Copyright © 2012–2019, Texas Instruments Incorporated

t_q	100	ns	=	t_{CAN_CLK}
delay of bus driver	90	ns	=	
delay of receiver circuit	40	ns	=	
delay of bus line (40m)	220	ns	=	
t_{Prop}	700	ns	=	2*delays = 7 • t_q
t_{SJW}	100	ns	=	1 • t_q
t_{TSeg1}	800	ns	=	t_{Prop} + t_{SJW}
t_{TSeg2}	100	ns	=	Information Processing Time + 1 • t_q
$t_{Sync\text{-}Seg}$	100	ns	=	1 • t_q
bit time	1000	ns	=	$t_{Sync\text{-}Seg}$ + t_{TSeg1} + t_{TSeg2}
tolerance for CAN_CLK	0.43	%	=	$\frac{\min(TSeg1, Tseg2)}{2x\ (13x(bit\_time\text{-}TSeg2))}$
			=	$\frac{0{,}1\mu s}{2x)13x(\ 1\mu s\ 0{,}1\mu s))}$

In this example, the concatenated bit time parameters are $(1\text{-}1)_3$&$(8\text{-}1)_4$&$(1\text{-}1)_2$&$(1\text{-}1)_6$, so the Bit Timing register is programmed to = 0x00000700.

23.12.2.3 Example for Bit Timing at low Baudrate

In this example, the frequency of CAN_CLK is 2 MHz, BRP is 1, the bit rate is 100 KBit/s.

t_q	1	µs	=	2 • t_{CAN_CLK}
delay of bus driver	200	ns	=	
delay of receiver circuit	80	ns	=	
delay of bus line (40m)	220	ns	=	
t_{Prop}	1	µs	=	1 • t_q
t_{SJW}	4	µs	=	4 • t_q
t_{TSeg1}	5	µs	=	t_{Prop} + t_{SJW}
t_{TSeg2}	4	µs	=	Information Processing Time + 4 • t_q
$t_{Sync\text{-}Seg}$	1	µs	=	1 • t_q
bit time	10	µs	=	$t_{Sync\text{-}Seg}$ + t_{TSeg1} + t_{TSeg2}
tolerance for CAN_CLK	3.08	%	=	$\frac{\min(TSeg1, TSeg2)}{2x(13x(bit\_time\text{-}TSeg2))}$
			=	$\frac{4\mu s}{2x(13x(9\mu s-4\mu s))}$

In this example, the concatenated bit time parameters are $(4\text{-}1)_3$&$(5\text{-}1)_4$&$(4\text{-}1)_2$&$(2\text{-}1)_6$, so the Bit Timing register is programmed to = 0x000034C1.

23.13 Message Interface Register Sets

The interface register sets control the CPU read and write accesses to the Message RAM. There are two interface register sets for read or write access (IF1 and IF2) and one Interface Register Set for read access only (IF3).

Copyright © 2012–2019, Texas Instruments Incorporated

Due to the structure of the Message RAM, it is not possible to change single bits or bytes of a message object. Instead, always a complete message object in the Message RAM is accessed. Therefore, the data transfer from the IF1 or IF2 registers to the Message RAM requires the message handler to perform a read-modify-write cycle: first those parts of the message object that are not to be changed are read from the Message RAM into the Interface Register Set, and after the update, the whole content of the Interface Register Set is written into the message object.

After the partial write of a message object, those parts of the Interface register set which are not selected in the Command register, will be set to the actual contents of the selected message object. After the partial read of a message object, those parts of the Interface Register Set which are not selected in the Command register, will be left unchanged.

By buffering the data to be transferred, the Interface Register sets avoid conflicts between concurrent CPU accesses to the Message RAM and CAN message reception and transmission. A complete message object (see Section 23.14.1) or parts of the message object may be transferred between the Message RAM and the IF1 or IF2 register set (see Section 23.15.15) in one single transfer. This transfer, performed in parallel on all selected parts of the message object, guarantees the data consistency of the CAN message.

That being said, there is one condition that can cause a write access to the message RAM to be lost. If MsgVal = 1 for the message object which is accessed and CAN communication is ongoing, a transfer from the IFx register to message RAM may be lost. The reason for this is that it might happen that the IFx register write to the message RAM occurs in between a read-modify-write access of the Host Message Handler when it is in the process of receiving a message for the same message object.

To avoid this issue with receive mail boxes, reset MsgVal before changing any of the following: Id28-0, Xtd, Dir, DLC3-0, RxIE, TxIE, RmtEn, EoB, Umask, Msk28-0, MXtd, and MDir.

To avoid this issue with transmit mail boxes, reset MsgVal before changing any of the following: Dir, RxIE, TxIE, RmtEn, EoB, Umask, Msk28-0, MXtd, and MDir. Other fields not listed above, like Data, may be changed without fear of losing a write to the message RAM.

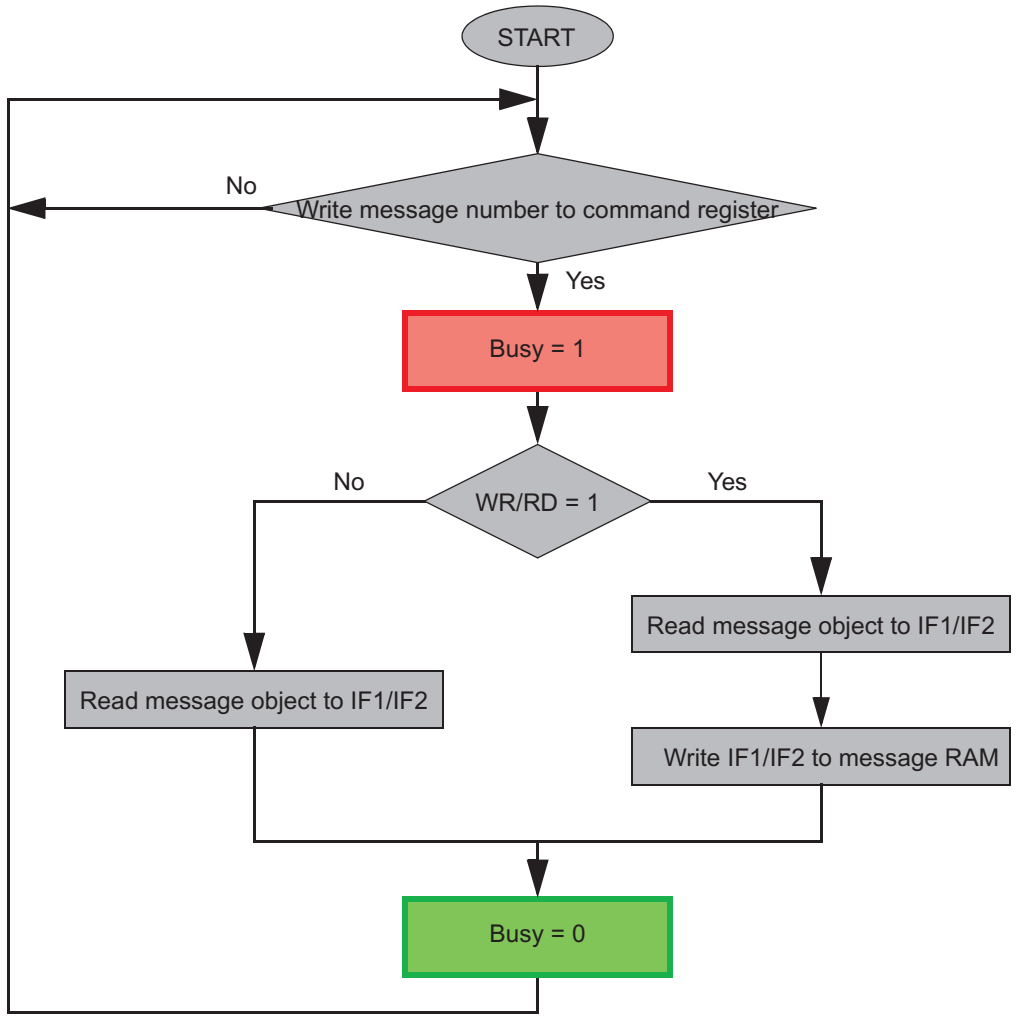
23.13.1 Message Interface Register Sets 1 and 2

The IF1 and IF2 register sets control the data transfer to and from the message object. The Command register addresses the desired message object in the Message RAM and specifies whether a complete message object or only parts should be transferred. The data transfer is initiated by writing the message number to the bits [7:0] of the Command register.

When the CPU initiates a data transfer between the IF1 or IF2 registers and Message RAM, the message handler sets the Busy bit in the respective Command register to '1'. After the transfer has completed, the Busy bit is set back to '0' (see Figure 23-16).

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 23-16. Data Transfer Between IF1 and IF2 Registers and Message RAM



23.13.2 IF3 Register Set

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU. The automatic update functionality can be programmed for each message object (see IF3 Update Enable register, Section 23.15.24).

All valid message objects in Message RAM which are configured for automatic update, will be checked for active NewDat flags. If such a message object is found, it will be transferred to the IF3 register, controlled by IF3 Observation register. If more than one NewDat flag is active, the message object with the lowest number has the highest priority for automatic IF3 update.

The NewDat bit in the message object will be reset by a transfer to IF3.

If CAN internal IF3 update is complete, an IF3 interrupt can also be generated.

NOTE: The IF3 register set cannot be used for transferring data into message objects.

23.14 Message RAM

The CAN Message RAM contains message objects and parity bits for the message objects. There are 32 message objects in the Message RAM.

Copyright © 2012–2019, Texas Instruments Incorporated

During normal operation, accesses to the Message RAM are performed via the Interface Register sets, and the CPU cannot directly access the Message RAM.

The Interface register sets IF1 and IF2, provide indirect read/write access from the CPU to the Message RAM. The IF1 and IF2 register sets can buffer control and user data to be transferred to and from the message objects.

The third Interface register set IF3 can be configured to automatically receive control and user data from the Message RAM when a message object has been updated after reception of a CAN message. The CPU does not need to initiate the transfer from Message RAM to the IF3 register set.

The message handler avoids potential conflicts between concurrent accesses to Message RAM and CAN frame reception/transmission.

The message RAM can only be accessed in debug mode. The message RAM base address is 0x2000 above the base address of the CAN peripheral.

23.14.1 Structure of Message Objects

Figure 23-17 shows the structure of a message object.

The grayed fields are those parts of the message object which are represented in dedicated registers. For example, the transmit request flags of all message objects are represented in centralized transmit request registers.

Figure 23-17. Structure of a Message Object

Message Object												
UMask	Msk[28:0]	MXtd	MDir	EoB	unused	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID[28:0]	Xtd	Dir	DLC[3:0]	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7

Table 23-2. Message Object Field Descriptions

Name	Value	Description
MsgVal		Message valid
	0	The message object is ignored by the message handler.
	1	The message object is to be used by the message handler.
		Note: This bit may be kept at level '1' even when the identifier bits ID[28:0], the control bits Xtd, Dir, or the data length code DLC[3:0] are changed. It should be reset if the Messages Object is no longer required.
UMask		Use Acceptance Mask
	0	Mask bits (Msk[28:0], MXtd and MDir) are ignored and not used for acceptance filtering.
	1	Mask bits are used for acceptance filtering.
		Note: If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
ID[28:0]		Message Identifier
	ID[28:0]	29-bit ("extended") identifier bits
	ID[28:18]	11-bit ("standard") identifier bits
Msk[28:0]		Identifier Mask
	0	The corresponding bit in the message identifier is not used for acceptance filtering (don't care).
	1	The corresponding bit in the message identifier is used for acceptance filtering.
		Note: The bit functionality in the DCAN module is the opposite of the "Local Acceptance Mask" bit functionality in the eCAN module found in older C28xx devices, where a "1" means the corresponding bit is NOT used for filtering and "0" means it is used.
Xtd		Extended Identifier
	0	The 11-bit ("standard") identifier will be used for this message object.
	1	The 29-bit ("extended") identifier will be used for this message object.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-2. Message Object Field Descriptions (continued)

Name	Value	Description
MXtd		Mask Extended Identifier
	0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
	1	The extended identifier bit (IDE) is used for acceptance filtering.
		Note: When 11-bit ("standard") Identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.
Dir		Message Direction
	0	Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, the message is stored in this message object.
	1	Direction = transmit: On TxRqst, a data frame is transmitted. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one).
MDir		Mask Message Direction
	0	The message direction bit (Dir) has no effect on the acceptance filtering.
	1	The message direction bit (Dir) is used for acceptance filtering.
EOB		End of Block
	0	The message object is part of a FIFO Buffer block and is not the last message object of this FIFO Buffer block.
	1	The message object is a single message object or the last message object in a FIFO Buffer Block.
		Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.
NewDat		New Data
	0	No new data has been written into the data bytes of this message object by the message handler since the last time when this flag was cleared by the CPU.
	1	The message handler or the CPU has written new data into the data bytes of this message object.
MsgLst		Message Lost (only valid for Message Objects with direction = receive)
	0	No message was lost since the last time when this bit was reset by the CPU.
	1	The message handler stored a new message into this message object when NewDat was still set, so the previous message has been overwritten.
RxIE		Receive Interrupt Enable
	0	IntPnd will not be triggered after the successful reception of a frame.
	1	IntPnd will be triggered after the successful reception of a frame.
TxIE		Transmit Interrupt Enable
	0	IntPnd will not be triggered after the successful transmission of a frame.
	1	IntPnd will be triggered after the successful transmission of a frame.
IntPnd		Interrupt Pending
	0	This message object is not the source of an interrupt. This message object is the source of an interrupt.
	1	The Interrupt Identifier in the Interrupt register will point to this message object if there is no other interrupt source with higher priority.
RmtEn		Remote Enable
	0	At the reception of a remote frame, TxRqst is not changed.
	1	At the reception of a remote frame, TxRqst is set.
		Note: See Section 23.11.8 for details on the setup of RmtEn and UMask for remote frames.
TxRqst		Transmit Request
	0	This message object is not waiting for a transmission.
	1	The transmission of this message object is requested and is not yet done.
DLC[3:0]		Data length code
	0-8	Data frame has 0-8 data bytes.
	8-15	Data frame has 8 data bytes.
		Note: The data length code of a message object must be defined to the same value as in the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-2. Message Object Field Descriptions (continued)

Name	Value	Description
Data 0		1st data byte of a CAN data frame
Data 1		2nd data byte of a CAN data frame
Data 2		3rd data byte of a CAN data frame
Data 3		4th data byte of a CAN data frame
Data 4		5th data byte of a CAN data frame
Data 5		6th data byte of a CAN data frame
Data 6		7th data byte of a CAN data frame
Data 7		8th data byte of a CAN data frame
		Note: Byte Data 0 is the first data byte shifted into the shift register of the CAN Core during a reception, byte Data 7 is the last. When the message handler stores a data frame, it will write all the eight data bytes into a message object. If the data length code is less than 8, the remaining bytes of the message object may be overwritten by undefined values.

23.14.2 Addressing Message Objects in RAM

The starting location of a particular message object in RAM is:

Message RAM base address + (message object number) * 0x20.

This means that Message Object 1 starts at offset 0x0020; Message Object 2 starts at offset 0x0040, etc.

NOTE: '0' is not a valid message object number. At address 0x0000, the last message object (32) (with the lowest priority) is located. Writing to the address of an unimplemented message object may overwrite an implemented message object.

Message Object number 1 has the highest priority.

Table 23-3. Message RAM Addressing in Debug Mode

Message Object Number	Offset From Base Address	Word Number	Debug Mode[(1)]
last implemented (here:32)	0x0000	1	Parity
	0x0004	2	MXtd,MDir,Mask
	0x0008	3	Xtd,Dir,ID
	0x000C	4	Ctrl
	0x0010	5	Data Bytes 3-0
	0x0014	6	Data Bytes 7-4
1	0x0020	1	Parity
	0x0024	2	MXtd,MDir,Mask
	0x0028	3	Xtd,Dir,ID
	0x002C	4	Ctrl
	0x0030	5	Data Bytes 3-0
	0x0034	6	Data Bytes 7-4
2	0x0040	1	Parity
	0x0044	2	MXtd,MDir,Mask
	0x0048	3	Xtd,Dir,ID
	0x004C	4	Ctrl
	0x0050	5	Data Bytes 3-0
	0x0054	6	Data Bytes 7-4
…	…	…	…

(1) See Section 23.14.3.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-3. Message RAM Addressing in Debug Mode (continued)

Message Object Number	Offset From Base Address	Word Number	Debug Mode[1]
31	0x03E0	1	Parity
	0x03E4	2	MXtd,MDir,Mask
	0x03E8	3	Xtd,Dir,ID
	0x03EC	4	Ctrl
	0x03F0	5	Data Bytes 3-0
	0x03F4	6	Data Bytes 7-4

23.14.3 Message RAM Representation in Debug Mode

In debug mode, the Message RAM will be memory mapped. This allows the external debug unit to access the Message RAM.

NOTE: During debug mode, the Message RAM cannot be accessed via the IFx register sets.

Figure 23-18. Message RAM Representation in Debug Mode

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

MsgAddr + 0x00

Reserved															
Reserved														Parity[4:0]	

MsgAddr + 0x04

MXtd	MDir	Rsvd	Msk[28:16]												
Msk[15:0]															

MsgAddr + 0x08

Rsvd	Xtd	Dir	ID[28:16]												
ID[15:0]															

MsgAddr + 0x0C

Reserved															
Rsvd	MsgLst	Rsvd	UMask	TxIE	RxIE	RmtEn	Rsvd	EOB	Reserved					DLC[3:0]	

MsgAddr + 0x10

Data 3								Data 2							
Data 1								Data 0							

MsgAddr + 0x14

Data 7								Data 6							
Data 5								Data 4							

23.15 CAN Control Registers

The base address for the CAN0 registers is 0x4007 0000 and the base address for the CAN1 registers is 0x4007 4000.

Table 23-4. CAN Control Registers

Offset	Acronym	Register Description	See
0x00	CAN CTL	CAN Control Register	Section 23.15.1
0x04	CAN ES	Error and Status Register	Section 23.15.2
0x08	CAN ERRC	Error Counter Register	Section 23.15.3
0x0C	CAN BTR	Bit Timing Register	Section 23.15.4
0x10	CAN INT	Interrupt Register	Section 23.15.5

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-4. CAN Control Registers (continued)

Offset	Acronym	Register Description	See
0x14	CAN TEST	Test Register	Section 23.15.6
0x1C	CAN PERR	Parity Error Code Register	Section 23.15.7
0x80	CAN ABOTR	Auto-Bus-On Time Register	Section 23.15.8
0x88	CAN TXRQ	Transmission Request Register	Section 23.15.9
0x9C	CAN NWDAT	New Data Register	Section 23.15.10
0xB0	CAN INTPND	Interrupt Pending Register	Section 23.15.11
0xC4	CAN MSGVAL	Message Valid Register	Section 23.15.12
0xD8	CAN INTMUX	Interrupt Multiplexer Register	Section 23.15.13
0x100	CAN IF1CMD	IF1 Command Register	Section 23.15.14
0x104	CAN IF1MSK	IF1 Mask Register	Section 23.15.15
0x108	CAN IF1ARB	IF1 Arbitration Register	Section 23.15.16
0x10C	CAN IF1MCTL	IF1 Message Control Register	Section 23.15.17
0x110	CAN IF1DATA	IF1 Data A Register	Section 23.15.18
0x114	CAN IF1DATB	IF1 Data B Register	Section 23.15.18
0x120	CAN IF2CMD	IF2 Command Register	Section 23.15.14
0x124	CAN IF2MSK	IF2 Mask Register	Section 23.15.15
0x128	CAN IF2ARB	IF2 Arbitration Register	Section 23.15.16
0x12C	CAN IF2MCTL	IF2 Message Control Register	Section 23.15.17
0x130	CAN IF2DATA	IF2 Data A Register	Section 23.15.18
0x134	CAN IF2DATB	IF2 Data B Register	Section 23.15.18
0x140	CAN IF3OBS	IF3 Observation Register	Section 23.15.19
0x144	CAN IF3MSK	IF3 Mask Register	Section 23.15.20
0x148	CAN IF3ARB	IF3 Arbitration Register	Section 23.15.21
0x14C	CAN IF3MCTL	IF3 Message Control Register	Section 23.15.22
0x150	CAN IF3DATA	IF3 Data A Register	Section 23.15.23
0x154	CAN IF3DATB	IF3 Data B Register	Section 23.15.23
0x160	CAN IF3UPD	IF3 Update Enable Register	Section 23.15.24

After hardware reset, the registers of the CAN hold the values shown in the register descriptions.

Additionally, the bus-off state is reset and the CAN_TX pin is set to recessive (HIGH). The Init bit in the CAN Control register is set to enable the software initialization. The CAN will not influence the CAN bus until the CPU resets Init to '0'.

23.15.1 CAN Control Register (CAN CTL)

The CAN Control register (CAN CTL) is shown and described in the figure and table below.

Figure 23-19. CAN Control Register (CAN CTL) [offset = 0x00]

31	26	25	24	23	18	17	16
Reserved		WUBA	PDR	Reserved		IE1	InitDbg
R-0		R/W-0	R/W-0	R-0		R/W-0	R-0

15	14	13	10	9	8	7	6	5	4	3	2	1	0
SWR	Rsvd	PMD		ABO	IDS	Test	CCE	DAR	Rsvd	EIE	SIE	IE0	Init
R/WP-0	R-0	R/W-0x5		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1

LEGEND: R = Read; R/W = Read/Write; WP = Write Protected by Init bit; *-n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-5. CAN Control Register (CAN CTL) Field Descriptions

Bit	Name	Value	Description
31-26	Reserved		Reserved
25	WUBA		Automatic wake up on bus activity enable bit. This bit is used to enable/disable Automatic wake up on bus activity, when in local power down mode.
		0	No detection of a dominant CAN bus level while in local power down mode.
		1	Detection of a dominant CAN bus level while in local power down mode is enabled. On occurrence of a dominant CAN bus level, the wake up sequence is started. **Note:** The CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power down and automatic wake-up mode, will be lost.
24	PDR		Power Down Mode Request bit. This bit is used to put the CAN module in local power down mode.
		0	No application request for local low power down mode. If the application has cleared this bit while CAN is in power down mode, the INIT bit has to be cleared as well.
		1	Power down mode has been requested by application. The CAN module will acknowledge this mode by setting the PDA bit in Error and Status Register. The local clocks will be turned off by CAN internal logic.
23-18	Reserved		Reserved
17	IE1		Interrupt line 1 Enable
		0	CAN1INT is disabled.
			CAN1INT is enabled. Interrupts will assert line CANINT1 to 1; line remains active until pending interrupts are processed.
16	InitDbg		Internal init state while debug access
		0	Not in debug mode, or debug mode requested but not entered.
		1	Debug mode requested and internally entered; the CAN is ready for debug accesses.
15	SWR		Software Reset Enable bit. This bit activates the software reset.
		0	Normal operation
		1	Module is forced to reset state. This bit will get cleared automatically one clock cycle after execution of software reset.
			Note: To execute software reset the following procedure is necessary: 1. Set INIT bit to shut down CAN communication. 2. Set SWR bit . This bit is EALLOW protected. Note: This bit is write-protected by the INIT bit.
14	Reserved		Reserved
13-10	PMD		Parity on/off
		0101	0101 Parity function disabled
		xxxx	x(any other value) Parity function enabled
9	ABO		Auto-Bus-On Enable
		0	Auto-Bus-On feature is disabled
		1	Auto-Bus-On feature is enabled
8	IDS		Interruption Debug Support Enable
		0	When Debug mode is requested, CAN will wait for a started transmission or reception to be completed before entering Debug mode.
		1	When Debug mode is requested, CAN will interrupt any transmission or reception, and enter Debug mode immediately.
7	Test		Test Mode Enable
		0	Disable Test mode (normal operation)
		1	Enable Test mode
6	CCE		Configuration Change Enable
		0	The CPU has no write access to the configuration registers.
		1	The CPU has write access to the configuration registers (when Init bit is set).
5	DAR		Disable Automatic Retransmission
		0	Automatic retransmission of "not successful" messages enabled.
		1	Automatic retransmission disabled.
4	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-5. CAN Control Register (CAN CTL) Field Descriptions (continued)

Bit	Name	Value	Description
3	EIE		Error Interrupt Enable
		0	Disabled - PER, BOff and EWarn bits cannot generate an interrupt.
		1	Enabled - PER, BOff and EWarn bits can generate an interrupt at CAN0INT line and affect the Interrupt Register.
2	SIE		Status Change Interrupt Enable
		0	Disabled - WakeUpPnd, RxOk, TxOk and LEC bits cannot generate an interrupt.
		1	Enabled - WakeUpPnd, RxOk, TxOk and LEC can generate an interrupt at CAN0INT line and affect the Interrupt Register.
1	IE0		Interrupt line 0 Enable
		0	Disabled - Module Interrupt CAN0INT is always low.
		1	Enabled - Interrupts will assert line CAN0INT to one; line remains active until pending interrupts are processed.
0	Init		Initialization mode bit. This bit may be set by software, a hardware reset, or the bus-off condition.
		0	Normal operation.
		1	Initialization mode is entered.

NOTE: The Bus-Off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting Init bit. If the module goes Bus-Off, it will automatically set the Init bit and stop all bus activities. When the Init bit is cleared by the application again, the module will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus-Off recovery sequence, the error counters will be reset. After the Init bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 Error code is written to the Error and Status Register, enabling the CPU to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the proceeding of the Bus-Off recovery sequence.

23.15.2 Error and Status Register (CAN ES)

The Error and Status register (CAN ES) is shown and described in the figure and table below.

Figure 23-20. Error and Status Register (CAN ES) [offset = 0x04]

31	16
Reserved	
R-0	

15	10	9	8	7	6	5	4	3	2	0
Reserved		Wake UpPnd	PER	BOff	EWarn	EPass	RxOK	TxOK	LEC	
R-0		R/C-0	R/C-0	R-0	R-0	R-0	R/C-0	R/C-0	R/S-111	

LEGEND: R = Read; S = Set by Read; C = Clear by Read; -*n* = value after reset

Table 23-6. Error and Status Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Reserved
9	WakeUpPnd		Wake Up Pending This bit can be used by the CPU to identify the CAN as the source to wake up the system.
		0	No Wake Up is requested by CAN.
		1	Wake up is requested. CAN has initiated a wake up of the system due to dominant CAN bus while module power down. This bit will be reset if Error and Status Register is read.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-6. Error and Status Register Field Descriptions (continued)

Bit	Field	Value	Description
8	PER		Parity Error Detected
		0	No parity error has been detected since last read access.
		1	The parity check mechanism has detected a parity error in the Message RAM. This bit will be reset if Error and Status Register is read.
7	BOff		Bus-Off State
		0	The CAN module is not in Bus-Off state.
		1	The CAN module is in Bus-Off state.
6	EWarn		Warning State
		0	Both error counters are below the error warning limit of 96.
		1	At least one of the error counters has reached the error warning limit of 96.
5	EPass		Error Passive State
		0	On CAN Bus error, the CAN could send active error frames.
		1	The CAN Core is in the error passive state as defined in the CAN Specification.
4	RxOk		Received a message successfully
		0	No message has been successfully received since the last time when this bit was read by the CPU. This bit is never reset by CAN internal events.
		1	A message has been successfully received since the last time when this bit was reset by a read access of the CPU (independent of the result of acceptance filtering). This bit will be reset if Error and Status Register is read.
3	TxOk		Transmitted a message successfully
		0	No message has been successfully transmitted since the last time when this bit was read by the CPU. This bit is never reset by CAN internal events.
		1	A message has been successfully transmitted (error free and acknowledged by at least one other node) since the last time when this bit was reset by a read access of the CPU. This bit will be reset if Error and Status Register is read.
2-0	LEC		Last Error Code The LEC field indicates the type of the last error on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error.
		0	No Error
		1	Stuff Error: More than five equal bits in a row have been detected in a part of a received message where this is not allowed.
		2	Form Error: A fixed format part of a received frame has the wrong format.
		3	Ack Error: The message this CAN Core transmitted was not acknowledged by another node.
		4	Bit1 Error: During the transmission of a message (with the excep-tion of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.
		5	Bit0 Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value '0'), but the monitored bus level was recessive. During Bus-Off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus-Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
		6	CRC Error: In a received message, the CRC check sum was incorrect. (CRC received for an incoming message does not match the calculated CRC for the received data).
		7	No CAN bus event was detected since the last time when CPU has read the Error and Status Register. Any read access to the Error and Status Register re-initializes the LEC to value '7'.

Interrupts are generated by bits PER, BOff and EWarn (if EIE bit in CAN Control Register is set) and by bits WakeUpPnd, RxOk, TxOk, and LEC (if SIE bit in CAN Control Register is set).

A change of bit EPass will not generate an Interrupt.

NOTE: Reading the Error and Status Register clears the WakeUpPnd, PER, RxOk and TxOk bits and sets the LEC to value '7'. Additionally, the Status Interrupt value (0x8000) in the Interrupt Register will be replaced by the next lower priority interrupt value.

Copyright © 2012–2019, Texas Instruments Incorporated

NOTE: For debug support, the auto clear functionality of Error and Status Register (clear of status flags by read) is disabled when in Debug mode.

23.15.3 Error Counter Register (CAN ERRC)

The Error Counter register (CAN ERRC) is shown and described in the figure and table below.

Figure 23-21. Error Counter Register (CAN ERRC) [offset = 0x08]

31	16
Reserved	
R-0	

15	14	8	7	0
RP	REC[6:0]		TEC[7:0]	
R-0	R-0		R-0	

LEGEND: R = Read; -*n* = value after reset

Table 23-7. Error Counter Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	RP		Receive Error Passive
		0	The Receive Error Counter is below the error passive level.
		1	The Receive Error Counter has reached the error passive level as defined in the CAN Specification.
14-8	REC[6:0]		Receive Error Counter. Actual state of the Receive Error Counter (values from 0 to 255).
7-0	TEC[7:0]		Transmit Error Counter. Actual state of the Transmit Error Counter (values from 0 to 255).

23.15.4 Bit Timing Register (CAN BTR)

The Bit Timing register (CAN BTR) is shown and described in the figure and table below.

Figure 23-22. Bit Timing Register (CAN BTR) [offset = 0x0C]

31	20	19	16
Reserved		BRPE	
R-0		R/WP-0	

15	14	12	11	8	7	6	5	0
Rsvd	TSeg2		TSeg1		SJW		BRP	
R-0	R/WP-0x2		R/WP-0x3		R/WP-0		R/WP-0x1	

LEGEND: R = Read; WP = Write Protected by CCE bit; -*n* = value after reset

Table 23-8. Bit Timing Register Field Descriptions

Bit	Field	Value	Description
31-20	Reserved		Reserved
19-16	BRPE	0x00-0x0F	Baud Rate Prescaler Extension. Valid programmed values are 0 to 15. By programming BRPE the Baud Rate Prescaler can be extended to values up to 1024.
15	Reserved		Reserved
14-12	TSeg2	0x0-0x7	Time segment after the sample point Valid programmed values are 0 to 7. The actual TSeg2 value which is interpreted for the Bit Timing will be the programmed TSeg2 value + 1.
11-8	TSeg1	0x01-0x0F	Time segment before the sample point Valid programmed values are 1 to 15. The actual TSeg1 value interpreted for the Bit Timing will be the programmed TSeg1 value + 1.
7-6	SJW	0x0-0x3	Synchronization Jump Width Valid programmed values are 0 to 3. The actual SJW value interpreted for the Synchronization will be the programmed SJW value + 1.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-8. Bit Timing Register Field Descriptions (continued)

Bit	Field	Value	Description
5-0	BRP	0x00-0x3F	Baud Rate Prescaler Value by which the CAN_CLK frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid programmed values are 0 to 63. The actual BRP value interpreted for the Bit Timing will be the programmed BRP value + 1.

NOTE: This register is only writable if CCE and Init bits in the CAN Control Register are set.

NOTE: The CAN bit time may be programmed in the range of 8 to 25 time quanta.

NOTE: The CAN time quantum may be programmed in the range of 1 to1024 CAN_CLK periods.

With a CAN_CLK of 8 MHz and BRPE = 0x00, the reset value of 0x00002301 configures the CAN for a bit rate of 500kBit/s.

For details see Section 23.12.

23.15.5 Interrupt Register (CAN INT)

The Interrupt register (CAN INT) is shown and described in the figure and table below.

Figure 23-23. Interrupt Register (CAN INT) [offset = 0x10]

31	24	23	16
Reserved		Int1ID[7:0]	
R-0		R-0	

15	0
Int0ID[15:0]	
R-0	

LEGEND: R = Read; -*n* = value after reset

Table 23-9. Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23-16	INT1ID		Interrupt 1 Identifier (indicates the message object with the highest pending interrupt)
		0x00	No interrupt is pending
		0x01-0x20	Number of the mailbox which caused the interrupt.
		0x21-0xFF	Unused
			If several interrupts are pending, the CAN Interrupt register will point to the pending interrupt with the highest priority. The CAN1INT interrupt line remains active until INT1ID reaches value 0 (the cause of the interrupt is reset) or until IE1 is cleared.
			A message interrupt is cleared by clearing the mailbox's IntPnd bit.
			Among the message interrupts, the mailbox's interrupt priority decreases with increasing message number.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-9. Descriptions (continued)

Bit	Field	Value	Description
15-0	INT0ID		Interrupt Identifier 0. This bit field indicates the source of the interrupt.
		0x0000	No interrupt is pending
		0x0001-0x020	Number of the mailbox which caused the interrupt.
		0x0021-0x7FFF	Unused
		0x8000	Error and Status register value is not 0x07.
		0x8001-0xFFFF	Unused
			If several interrupts are pending, the CAN Interrupt register will point to the pending interrupt with the highest priority. The CAN0INT interrupt line remains active until INT0ID reaches value 0 (the cause of the interrupt is reset) or until IE0 is cleared. The status interrupt has the highest priority.
			A message interrupt is cleared by clearing the mailbox's IntPnd bit.
			Among the message interrupts, the mailbox's interrupt priority decreases with increasing message number.

23.15.6 Test Register (CAN TEST)

The Test register (CAN TEST) is shown and described in the figure and table below.

Figure 23-24. Test Register (CAN TEST) [offset = 0x14]

31										16
Reserved										
R-0										
15	10	9	8	7	6	5	4	3	2	0
Reserved		RDA	EXL	Rx	Tx[1:0]		LBack	Silent	Reserved	
R-0		R/WP-0	R/WP-0	R-U	R/WP-0		R/WP-0	R/WP-0	R-0	

LEGEND: R = Read; WP = Write Protected by Test bit; -*n* = value after reset; -U = Undefined

Table 23-10. Test Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reserved
8	EXL		External Loopback Mode
		0	Disabled
		1	Enabled
7	Rx		Receive Pin. Monitors the actual value of the CAN_RX pin
		0	The CAN bus is dominant
		1	The CAN bus is recessive
6-5	Tx[1:0]		Control of CAN_TX pin Normal operation
		00	CAN_TX is controlled by the CAN Core.
		01	Sample Point can be monitored at CAN_TX pin.
		10	CAN_TX pin drives a dominant value.
		11	CAN_TX pin drives a recessive value.
4	LBack		Loopback Mode
		0	Disabled
		1	Enabled
3	Silent		Silent Mode
		0	Disabled
		1	Enabled

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-10. Test Register Field Descriptions (continued)

Bit	Field	Value	Description
2-0	Reserved		Reserved

For all test modes, the Test bit in CAN Control register needs to be set to one. If Test bit is set, the EXL, Tx1, Tx0, LBack and Silent bits are writable. Bit Rx monitors the state of pin CAN_RX and therefore is only readable. All Test register functions are disabled when the Test bit is cleared.

NOTE: The Test register is only writable if Test bit in CAN Control register is set.

NOTE: Setting Tx[1:0] other than '00' will disturb message transfer.

NOTE: When the internal loopback mode is active (bit LBack is set), bit EXL will be ignored.

23.15.7 Parity Error Code Register (CAN PERR)

The Parity Error Code register (CAN PERR) is shown and described in the figure and table below.

Figure 23-25. Parity Error Code Register (CAN PERR) [offset = 0x1C]

31	16
Reserved	
R-0	

15	11	10	8	7	0
Reserved		Word Number		Message Number	
R-0		R-U		R-U	

LEGEND: R = Read; -*n* = value after reset; -U = Undefined

Table 23-11. Parity Error Code Register Field Descriptions

Bit	Field	Value	Description
31-11	Reserved		Reserved
10-8	Word Number	0x01-0x05	Word number where parity error has been detected RDA word number (1 to 5) of the message object (according to the Message RAM representation in RDA mode, see Section 23.14.3).
7-0	Message Number	0x01-0x80	Message object number where parity error has been detected

If a parity error is detected, the PER flag will be set in the Error and Status Register. This bit is not reset by the parity check mechanism; it must be reset by reading the Error and Status Register. In addition to the PER flag, the Parity Error Code Register will indicate the memory area where the parity error has been detected (message number and word number).

If more than one word with a parity error was detected, the highest word number with a parity error will be displayed.

After a parity error has been detected, the register will hold the last error code until power is removed.

23.15.8 Auto-Bus-On Time Register (CAN ABOTR)

The Auto-Bus-On Time register (CAN ABOTR) is shown and described in the figure and table below.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 23-26. Auto-Bus-On Time Register (CAN ABOTR) [offset = 0x80]

31	16
ABO Time[31:16]	
R/W-0	

15	0
ABO Time[15:0]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 23-12. Auto-Bus-On Time Register Field Descriptions

Bit	Field	Value	Description
31-0	ABO Time		Number of clock cycles before a Bus-Off recovery sequence is started by clearing the Init bit. "Clock" refers to the input clock to the CAN module. This function has to be enabled by setting bit ABO in CAN Control Register. The Auto-Bus-On timer is realized by a 32-bit counter which starts to count down to zero when the module goes Bus-Off. The counter will be reloaded with the preload value of the ABO Time register after this phase.

NOTE: On write access to the CAN Control register while Auto-Bus-On timer is running, the Auto-Bus-On procedure will be aborted.

NOTE: During Debug mode, running Auto-Bus-On timer will be paused.

23.15.9 Transmission Request Registers (CAN TXRQ)

These registers hold the TxRqst bits of the implemented message objects. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific message object can be set/reset by the CPU via the IF1 or IF2 Message Interface registers, or by the message handler after reception of a remote frame or after a successful transmission.

Figure 23-27. Transmission Request Register (CAN TXRQ) [offset = 0x88]

31	16
TxRqst[31:16]	
R-0	

15	0
TxRqst[15:0]	
R-0	

LEGEND: R = Read; -*n* = value after reset

NOTE: Bits 0 through 31 correspond to message object 1 through 32, respectively.

Table 23-13. Transmission Request Register Field Descriptions

Bit	Field	Value	Description
31-0	TxRqst[31:0]		Transmission Request Bits (for all message objects)
		0	No transmission has been requested for this message object.
		1	The transmission of this message object is requested and is not yet done.

Copyright © 2012–2019, Texas Instruments Incorporated

23.15.10 New Data Registers (CAN NWDAT)

These registers hold the NewDat bits of the implemented message objects. By reading out these bits, the CPU can check for new data in the message objects. The NewDat bit of a specific message object can be set/reset by the CPU via the IF1 or IF2 Interface register sets, or by the message handler after reception of a data frame or after a successful transmission.

Figure 23-28. New Data Register (CAN NWDAT) [offset = 0x9C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NewDat[31:16]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NewDat[15:0]															
R-0															

LEGEND: R = Read; -*n* = value after reset

NOTE: Bits 0 through 31 correspond to message object 1 through 32, respectively.

Table 23-14. New Data Registers Field Descriptions

Bit	Field	Value	Description
31-0	NewDat[31:0]		New Data Bits (for all message objects)
		0	No new data has been written into the data portion of this message object by the message handler since the last time when 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.

23.15.11 Interrupt Pending Registers (CAN INTPND)

These registers hold the IntPnd bits of the implemented message objects. By reading out these bits, the CPU can check for pending interrupts in the message objects. The IntPnd bit of a specific message object can be set/reset by the CPU via the IF1 or IF2 Interface register sets, or by the message handler after a reception or a successful transmission.

Figure 23-29. Interrupt Pending Register (CAN INTPND) [offset = 0xB0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IntPndReg[31:16]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntPndReg[15:0]															
R-0															

LEGEND: R = Read; -*n* = value after reset

NOTE: Bits 0 through 31 correspond to message object 1 through 32, respectively.

Table 23-15. Interrupt Pending Registers Field Descriptions

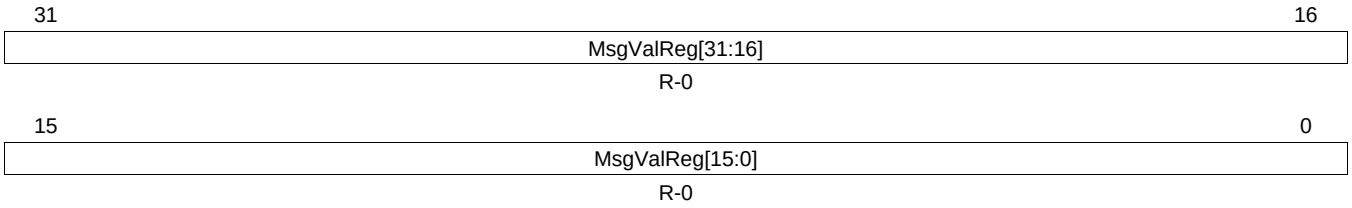
Bit	Field	Value	Description
31-0	IntPndReg[31:0]		Interrupt Pending Bits (for all message objects)
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt.

Copyright © 2012–2019, Texas Instruments Incorporated

23.15.12 Message Valid Registers (CAN MSGVAL)

These registers hold the MsgVal bits of the implemented message objects. By reading out these bits, the CPU can check which message objects are valid. The MsgVal bit of a specific message object can be set/reset by the CPU via the IF1 or IF2 Interface register sets, or by the message handler after a reception or a successful transmission.

Figure 23-30. Message Valid Register (CAN MSGVAL) [offset = 0xC4]



LEGEND: R = Read; -*n* = value after reset

NOTE: Bits 0 through 31 correspond to message object 1 through 32, respectively.

Table 23-16. Message Valid Registers Field Descriptions

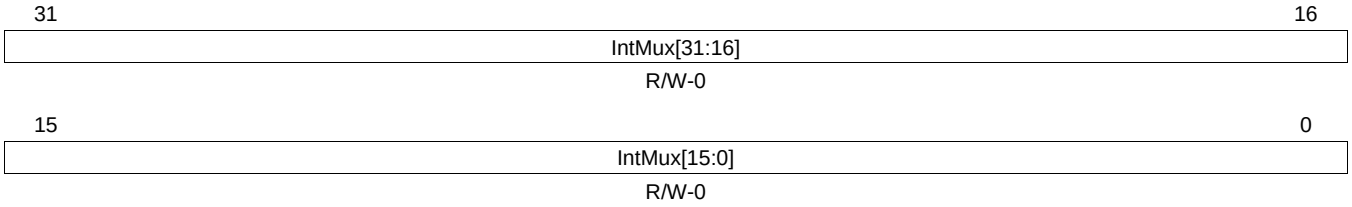
Bit	Field	Value	Description
31-0	MsgValReg[31:0]		Message Valid Bits (for all message objects)
		0	This message object is ignored by the message handler.
		1	This message object is configured and will be considered by the message handler.

23.15.13 Interrupt Multiplexer Registers (CAN INTMUX)

The IntMux flag determine for each message object, which of the two interrupt lines (CAN0INT or CAN1INT) will be asserted when the IntPnd of this message object is set. Both interrupt lines can be globally enabled or disabled by setting or clearing IE0 and IE1 bits in CAN Control Register.

The IntPnd bit of a specific message object can be set or reset by the CPU via the IF1 or IF2 Interface register sets, or by message handler after reception or successful transmission of a frame. This will also affect the Int0ID resp Int1ID flags in the Interrupt Register.

Figure 23-31. Interrupt Multiplexer Register (CAN INTMUX) [offset = 0xD8]



LEGEND: R/W = Read/Write; -*n* = value after reset

NOTE: Bit 0 in this register corresponds to message object 32, while bits 1 through 31 correspond to message objects 1 through 31 respectively.

Table 23-17. Interrupt Multiplexer Registers Field Descriptions

Bit	Field	Value	Description
31-0	IntMux[31:0]		Multiplexes IntPnd value to either CAN0INT or CAN1INT interrupt lines (for all message objects)
		0	CAN0INT line is active if corresponding IntPnd flag is one.
		1	CAN1INT line is active if corresponding IntPnd flag is one.

Copyright © 2012–2019, Texas Instruments Incorporated

23.15.14 IF1 and IF2 Command Registers (CAN IF1CMD, CAN IF2CMD)

The IF1 and IF2 Command registers configure and initiate the transfer between the IF1 or IF2 Register sets and the Message RAM. It is configurable which portions of the message object should be transferred.

A transfer is started when the CPU writes the message number to bits [7:0] of the IF1 or IF2 Command Register.

With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress. After 4 to 14 clock cycles, the transfer between the Interface Register and the Message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer concurs with a CAN message transmission, acceptance filtering, or message storage.

If the CPU writes to both IF1 and IF2 Command registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed. The following points must be borne in mind while writing to this register: (1) Do not write zeros to the whole register. (2) Write to the register in a single 32-bit write or write the upper 16-bits before writing to the lower 16- bits.

NOTE: While Busy bit is 1, IF1 and IF2 register sets are write protected.

NOTE: For debug support, the auto clear functionality of the IF1 or IF2 Command Registers is disabled during Debug mode.

NOTE: If an invalid Message Number is written to bits [7:0] of the IF1 or IF2 Command Register, the message handler may access an implemented (valid) message object instead.

Figure 23-32. IF1 Command Registers (CAN IF1CMD) [offset = 0x100]

31	24	23	22	21	20	19	18	17	16
Reserved		WR/RD	Mask	Arb	Control	Clr IntPnd	TxRqst / NewDat	Data A	Data B
R-0		R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

15	14	13	8	7	0
Busy	DMA active	Reserved		Message Number	
R-0	R/WP/C-0	R-0		R/WP-0x1	

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); C = Clear by IF1 access; -*n* = value after reset

Figure 23-33. IF2 Command Registers (CAN IF2CMD) [offset = 0x120]

31	24	23	22	21	20	19	18	17	16
Reserved		WR/RD	Mask	Arb	Control	Clr IntPnd	TxRqst / NewDat	Data A	Data B
R-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

15	14	13	6	5	0
Busy	DMA active	Reserved		Message Number	
R-0	R/WP/C-0	R-0		R/WP-0x1	

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); C = Clear by IF2 access; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-18. IF1 and IF2 Command Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23	WR/RD		Write/Read
		0	Direction = Read: Transfer direction is from the message object addressed by Message Number (Bits [7:0]) to the IF1 or IF2 register set.
		1	Direction = Write: Transfer direction is from the IF1 or IF2 register set to the message object addressed by Message Number (Bits [7:0])
22	Mask		Access Mask Bits
		0	Mask bits will not be changed
		1	Direction = Read: The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF1 or IF2 Register set.
			Direction = Write: The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the IF1 or IF2 register set to the message object addressed by Message Number (Bits [7:0]).
21	Arb		Access Arbitration Bits
		0	Arbitration bits will not be changed
		1	Direction = Read: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1 or IF2 register set.
			Direction = Write: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the IF1 or IF2 register set to the mes-sage object addressed by Message Number (Bits [7:0]).
20	Control		Access Control Bits
		0	Control bits will not be changed
		1	Direction = Read: The Message Control bits will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF1 or IF2 register set.
			Direction = Write: The Message Control bits will be transferred from the IF1 or IF2 register set to the message object addressed by Message Number (Bits [7:0]). If the TxRqst/NewDat bit in this register (Bit [18]) is set, the TxRqst/ NewDat bit in the IF1 or IF2 Message Control register will be ignored.
19	ClrIntPnd		Clear Interrupt Pending Bit
		0	IntPnd bit will not be changed
		1	Direction = Read: Clears IntPnd bit in the message object.
			Direction = Write: This bit is ignored. Copying of IntPnd flag from IF1 or IF2 registers to Message RAM can only be controlled by the Control flag (Bit [20]).
18	TxRqst/NewDat		Access Transmission Request Bit
		0	Direction = Read: NewDat bit will not be changed. Direction = Write: TxRqst/NewDat bit will be handled according to the Control bit.
		1	Direction = Read: Clears NewDat bit in the message object. Direction = Write: Sets TxRqst/NewDat in message object.
			Note: If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to one independent of the values in the IF1 or IF2 Message Control register.
			Note: A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1 or IF2 Message Control register always reflect the status before resetting them.
17	Data A		Access Data Bytes 0-3
		0	Data Bytes 0-3 will not be changed.
		1	Direction = Read: The Data Bytes 0-3 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF1 or IF2 register set.
			Direction = Write: The Data Bytes 0-3 will be transferred from the IF1 or IF2 register set to the message object addressed by the Message Number (Bits [7:0]).
			Note: The duration of the message transfer is independent of the number of bytes to be transferred.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-18. IF1 and IF2 Command Register Field Descriptions (continued)

Bit	Field	Value	Description
16	Data B		Access Data Bytes 4-7
		0	Data Bytes 4-7 will not be changed.
		1	Direction = Read: The Data Bytes 4-7 will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1 or IF2 register set.
			Direction = Write: The Data Bytes 4-7 will be transferred from the IF1 or F2 register set to the message object addressed by Message Number (Bits [7:0]).
			Note: The duration of the message transfer is independent of the number of bytes to be transferred.
15	Busy		Busy Flag
		0	No transfer between IF1 or IF2 register Set and Message RAM is in progress.
		1	Transfer between IF1 or IF2 register Set and Message RAM is in progress.
			This bit is set to one after the message number has been written to bits [7:0]. IF1 or IF2 register set will be write protected. The bit is cleared after read/write action has been finished.
14	DMAactive		Activation of DMA feature for subsequent internal IF1 and IF2 update
		0	DMA request line is independent of IF1 or IF2 activities.
		1	DMA is requested after completed transfer between IF1 or IF2 register set and Message RAM. The DMA request remains active until the first read or write to one of the IF1 or IF2 registers; an exception is a write to Message Number (Bits [7:0]) when DMAactive is one.
			Note: Due to the auto reset feature of the DMAactive bit, this bit has to be set for each subsequent DMA cycle separately.
13-8	Reserved		Reserved
7-0	Message Number		Number of message object in Message RAM which is used for data transfer
		0x00	Invalid message number
		0x01-0x20	Valid message numbers
		0x21-0xFF	Invalid message numbers

23.15.15 IF1 and IF2 Mask Registers (CAN IF1MSK, CAN IF2MSK)

The bits of the IF1 and IF2 Mask registers mirror the mask bits of a message object. The function of the relevant message objects bits is described in Section 23.14.1.

NOTE: While the Busy bit of the IF1 or IF2 Command Register is one, the IF1 or IF2 Register Set is write protected.

Figure 23-34. IF1 Mask Register (CAN IF1MSK) [offset = 0x104]

31	30	29	28 ... 16
MXtd	MDir	Rsvd	Msk[28:16]
R/WP-1	R/WP-1	R-1	R/WP-0x1FFF

15 ... 0
Msk[15:0]
R/WP-0xFFFF

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 23-35. IF2 Mask Register (CAN IF2MSK) [offset = 0x124]

31	30	29	28 ... 16
MXtd	MDir	Rsvd	Msk[28:16]
R/WP-1	R/WP-1	R-1	R/WP-0x1FFF

15 ... 0
Msk[15:0]
R/WP-0xFFFF

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Table 23-19. IF1 and IF2 Mask Registers Field Descriptions

Bit	Field	Value	Description
31	MXtd		Mask Extended Identifier
		0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
		1	The extended identifier bit (IDE) is used for acceptance filtering.
			When 11-bit ("standard") identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.
30	MDir		Mask Message Direction
		0	The message direction bit (Dir) has no effect on the acceptance filtering.
		1	The message direction bit (Dir) is used for acceptance filtering.
29	Reserved		Reserved
28-0	Msk[28:0]		Identifier Mask
		0	The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).
		1	The corresponding bit in the identifier of the message object is used for acceptance filtering.

23.15.16 IF1 and F2 Arbitration Registers (CAN IF1ARB, CAN IF2ARB)

The bits of the IF1 and IF2 arbitration registers mirror the arbitration bits of a message object. The function of the relevant message objects bits is described in Section 23.14.1.

NOTE: While Busy bit of the IF1 or IF2 Command Register is one, the IF1 or IF2 register set is write protected.

Figure 23-36. IF1 Arbitration Register (CAN IF1ARB) [offset = 0x108]

31	30	29	28 ... 16
MsgVal	Xtd	Dir	ID[28:16]
R/WP-0	R/WP-0	R/WP-0	R/WP-0

15 ... 0
ID[15:0]
R/WP-0

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 23-37. IF2 Arbitration Register (CAN IF2ARB) [offset = 0x128]

31	30	29	28	16
MsgVal	Xtd	Dir	ID[28:16]	
R/WP-0	R/WP-0	R/WP-0	R/WP-0	

15	0
ID[15:0]	
R/WP-0	

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Table 23-20. IF1 and IF2 Arbitration Registers Field Descriptions

Bit	Field	Value	Description
31	MsgVal		Message Valid
		0	The message object is ignored by the message handler.
		1	The message object is to be used by the message handler.
			The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required.
30	Xtd		Extended Identifier
		0	The 11-bit ("standard") Identifier is used for this message object.
		1	The 29-bit ("extended") Identifier is used for this message object.
29	Dir		Message Direction
		0	Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, that message is stored in this message object.
		1	Direction = transmit: On TxRqst, 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 = one).
28-0	ID[28:0]		Message Identifier
		ID[28:0]	29-bit Identifier ("Extended Frame")
		ID[28:18]	11-bit Identifier ("Standard Frame")

The Arbitration bits ID[28:0], Xtd, and Dir are used to define the identifier and type of outgoing messages and (together with the Mask bits Msk[28:0], MXtd, and MDir) for acceptance filtering of incoming messages.

A received message is stored into the valid message object with matching identifier and Direction = receive (data frame) or Direction = transmit (remote frame).

Extended frames can be stored only in message objects with Xtd = one, standard frames in message objects with Xtd = zero.

If a received message (data frame or remote frame) matches more than one valid message objects, it is stored into the one with the lowest message number.

23.15.17 IF1 and IF2 Message Control Registers (CAN IF1MCTL, CAN IF2MCTL)

The bits of the IF1 and IF2 Message Control registers mirror the message control bits of a message object. The function of the relevant message objects bits is described in Section 23.14.1.

NOTE: While the Busy bit of the IF1 and IF2 Command Register is one, IF1 and IF2 register sets are write protected.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 23-38. IF1 Message Control Register (CAN IF1MCTL) [offset = 0x10C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R-0			R/WP-0			

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 23-39. IF2 Message Control Register (CAN IF2MCTL) [offset = 0x12C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R-0			R/WP-0			

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Table 23-21. IF1 and IF2 Message Control Registers Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	NewDat		New Data
		0	No new data has been written into the data portion of this message object by the message handler since the last time when 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		Message Lost (only valid for message objects with direction = receive)
		0	No message lost since the last time when this bit was reset by the CPU.
		1	The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd		Interrupt Pending
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.
12	UMask		Use Acceptance Mask
		0	Mask ignored
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering
			If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
11	TxIE		Transmit Interrupt Enable
		0	IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
10	RxIE		Receive Interrupt Enable
		0	IntPnd will not be triggered after the successful reception of a frame.
		1	IntPnd will be triggered after the successful reception of a frame.
9	RmtEn		Remote Enable
		0	At the reception of a remote frame, TxRqst is not changed.
		1	At the reception of a remote frame, TxRqst is set.
			Note: See Section 23.11.8 for details on the setup of RmtEn and UMask for remote frames.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-21. IF1 and IF2 Message Control Registers Field Descriptions (continued)

Bit	Field	Value	Description
8	TxRqst		Transmit Request
		0	This message object is not waiting for a transmission.
		1	The transmission of this message object is requested and is not yet done.
7	EoB		End of Block
		0	The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.
		1	The message object is a single message object or the last message object in a FIFO Buffer Block.
			Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.
6-5	Reserved		Reserved
3-0	DLC[3:0]		Data length code
		0-8	Data frame has 0-8 data bits.
		9-15	data frame has 8 data bytes.
			Note: The data length code 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 will write the DLC to the value given by the received message.

23.15.18 IF1 and IF2 Data A and Data B Registers (CAN IF1DATA/DATB, CAN IF2DATA/DATB)

The data bytes of CAN messages are stored in the IF1 and IF2 registers in the following order. In a CAN data frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first

Figure 23-40. IF1 Data A Register (CAN IF1DATA) [offset = 0x110]

31	24	23	16
Data 3		Data 2	
R/WP-0		R/WP-0	

15	8	7	0
Data 1		Data 0	
R/WP-0		R/WP-0	

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 23-41. IF1 Data B Register (CAN IF1DATB) [offset = 0x114]

31	24	23	16
Data 7		Data 6	
R/WP-0		R/WP-0	

15	8	7	0
Data 5		Data 4	
R/WP-0		R/WP-0	

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 23-42. IF2 Data A Register (CAN IF2DATA) [offset = 0x130]

31	24	23	16
Data 3		Data 2	
R/WP-0		R/WP-0	

15	8	7	0
Data 1		Data 0	
R/WP-0		R/WP-0	

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 23-43. IF2 Data B Register (CAN IF2DATB) [offset = 0x134]

31	24	23	16
Data 7		Data 6	
R/WP-0		R/WP-0	

15	8	7	0
Data 5		Data 4	
R/WP-0		R/WP-0	

LEGEND: R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

23.15.19 IF3 Observation Register (CAN IF3OBS)

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU (see also Section 23.14.1).

The observation flags (Bits [4:0]) in the IF3 Observation register are used to determine, which data sections of the IF3 Interface Register set have to be read in order to complete a DMA read cycle. After all marked data sections are read, the CAN is enabled to update the IF3 Interface Register set with new data.

Any access order of single bytes or half-words is supported. When using byte or half-word accesses, a data section is marked as completed, if all bytes are read.

NOTE: If IF3 Update Enable is used and no Observation flag is set, the corresponding message objects will be copied to IF3 without activating the DMA request line and without waiting for DMA read accesses.

A write access to this register aborts a pending DMA cycle by resetting the DMA line and enables updating of IF3 Interface Register set with new data. To avoid data inconsistency, the DMA controller should be disabled before reconfiguring IF3 observation register.

Figure 23-44. IF3 Observation Register (CAN IF3OBS) [offset = 0x140]

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		DataB	DataA	Ctrl	Arb	Mask
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R = Read; R/W = Read/Write; -*n* = value after reset

Table 23-22. IF3 Observation Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4	DataB		Data B read observation
		0	Data B section has not to be read.
		1	Data B section has to be read to enable next IF3 update.
3	DataA		Data A read observation
		0	Data A section has not to be read.
		1	Data A section has to be read to enable next IF3 update.
2	Ctrl		Ctrl read observation
		0	Ctrl section has not to be read.
		1	Ctrl section has to be read to enable next IF3 update.
1	Arb		Arbitration data read observation
		0	Arbitration data has not to be read.
		1	Arbitration data has to be read to enable next IF3 update.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-22. IF3 Observation Register Field Descriptions (continued)

Bit	Field	Value	Description
0	Mask		Mask data read observation
		0	Mask data has not to be read.
		1	Mask data has to be read to enable next IF3 update.

23.15.20 IF3 Mask Register (CAN IF3MSK)

The IF3 Mask register (CAN IF3MSK) is shown and described in the figure and table below.

Figure 23-45. IF3 Mask Register (CAN IF3MSK) [offset = 0x144]

31	30	29	28	16
MXtd	MDir	Rsvd	Msk[28:16]	
R-1	R-1	R-1	R-0x1FFF	

15	0
Msk[15:0]	
R-0xFFFF	

LEGEND: R = Read; -*n* = value after reset

Table 23-23. IF1 and IF2 Mask Registers Field Descriptions

Bit	Field	Value	Description
31	MXtd		Mask Extended Identifier
		0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
		1	The extended identifier bit (IDE) is used for acceptance filtering.
			Note: When 11-bit ("standard") identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.
30	MDir		Mask Message Direction
		0	The message direction bit (Dir) has no effect on the acceptance filtering.
		1	The message direction bit (Dir) is used for acceptance filtering.
29	Reserved		Reserved
28-0	Msk[28:0]		Identifier Mask
		0	The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).
		1	The corresponding bit in the identifier of the message object is used for acceptance filtering.

23.15.21 IF3 Arbitration Register (CAN IF3ARB)

The IF3 arbitration register (CAN IF3ARB) is shown and described in the figure and table below.

Figure 23-46. IF3 Arbitration Register (CAN IF3ARB) [offset = 0x148]

31	30	29	28	16
MsgVal	Xtd	Dir	ID[28:16]	
R-0	R-0	R-0	R-0	

15	0
ID[15:0]	
R-0	

LEGEND: R = Read; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-24. IF3 Arbitration Register Field Descriptions

Bit	Field	Value	Description
31	MsgVal		Message Valid
		0	The message object is ignored by the message handler.
		1	The message object is to be used by the message handler.
			The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required.
30	Xtd		Extended Identifier
		0	The 11-bit ("standard") Identifier is used for this message object.
		1	The 29-bit ("extended") Identifier is used for this message object.
29	Dir		Message Direction
		0	Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, that message is stored in this message object.
		1	Direction = transmit: On TxRqst, 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 = one).
28-0	ID[28:0]		Message Identifier
		ID[28:0]	29-bit Identifier ("Extended Frame")
		ID[28:18]	11-bit Identifier ("Standard Frame")

23.15.22 IF3 Message Control Register (CAN IF3MCTL)

The IF3 Message Control register (CAN IF3MCTL) is shown and described in the figure and table below.

Figure 23-47. IF3 Message Control Register (CAN IF3MCTL) [offset = 0x14C]

31										16
Reserved										
R-0										

15	14	13	12	11	10	9	8	7	6 4	3 0
New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved	DLC[3:0]
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read; -*n* = value after reset

Table 23-25. IF3 Message Control Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reserved
15	NewDat		New Data
		0	No new data has been written into the data portion of this message object by the message handler since the last time when 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		Message Lost (only valid for message objects with direction = receive)
		0	No message lost since the last time when this bit was reset by the CPU.
		1	The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd		Interrupt Pending
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 23-25. IF3 Message Control Register Field Descriptions (continued)

Bit	Field	Value	Description
12	UMask		Use Acceptance Mask
		0	Mask ignored
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering
			If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
11	TxIE		Transmit Interrupt Enable
		0	IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
10	RxIE		Receive Interrupt Enable
		0	IntPnd will not be triggered after the successful reception of a frame.
		1	IntPnd will be triggered after the successful reception of a frame.
9	RmtEn		Remote Enable
		0	At the reception of a remote frame, TxRqst is not changed.
		1	At the reception of a remote frame, TxRqst is set.
			Note: See Section 23.11.8 for details on the setup of RmtEn and UMask for remote frames.
8	TxRqst		Transmit Request
		0	This message object is not waiting for a transmission.
		1	The transmission of this message object is requested and is not yet done.
7	EoB		End of Block
		0	The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.
		1	The message object is a single message object or the last message object in a FIFO Buffer Block.
			Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.
6-5	Reserved		Reserved
3-0	DLC[3:0]		Data length code
		0-8	Data frame has 0-8 data bytes.
		9-15	Data frame has 8 data bytes.
			Note: The data length code 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 will write the DLC to the value given by the received message.

23.15.23 IF3 Data A and Data B Registers (CAN IF3DATA/DATB)

The data bytes of CAN messages are stored in the IF3 registers in the following order: in a CAN data frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Figure 23-48. IF3 Data A Register (CAN IF3DATA) [offset = 0x150]

31	24	23	16
Data 3		Data 2	
R-0		R-0	
15	8	7	0
Data 1		Data 0	
R-0		R-0	

LEGEND: R = Read; -*n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 23-49. IF3 Data A Register (CAN IF3DATB) [offset = 0x154]

31	24	23	16
Data 7		Data 6	
R-0		R-0	

15	8	7	0
Data 5		Data 4	
R-0		R-0	

LEGEND: R = Read; -*n* = value after reset

23.15.24 IF3 Update Enable Registers (CAN IF3UPD)

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdE flag is set. This means that an active NewDat flag of this message object (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set.

NOTE: IF3 Update enable should not be set for transmit objects.

Figure 23-50. IF3 Update Enable Register (CAN IF3UPD) [offset = 0x160]

31	16
IF3UpdEn[32:17]	
R/W-0	

15	0
IF3UpdEn[16:1]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 23-26. IF3 Update Control Register Field Descriptions

Bit	Field	Value	Description
31-0	IF3UpdEn[31:0]		IF3 Update Enabled (for all message objects)
		0	Automatic IF3 update is disabled for this message object.
		1	Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.

Copyright © 2012–2019, Texas Instruments Incorporated

Cortex-M3 Processor

This chapter provides information on the implementation of the Cortex™-M3 processor, including the programming model, the memory model, the exception model, fault handling, and power management.

For technical details on the instruction set, see the *Cortex™-M3 Instruction Set Technical User's Manual*.

Topic **Page**

24.1 Overview **1568**
24.2 Block Diagram **1568**
24.3 Overview **1569**
24.4 Programming Model **1570**
24.5 Memory Model **1578**
24.6 Memory Regions, Types and Attributes **1579**
24.7 Exception Model **1584**
24.8 Fault Handling **1592**
24.9 Power Management **1593**
24.10 Instruction Set Summary **1595**

Copyright © 2012–2019, Texas Instruments Incorporated

24.1 Overview

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

- 32-bit ARM Cortex-M3 architecture optimized for small-footprint embedded applications
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set delivers the high performance expected of a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices, typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast multiplier
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Migration from the ARM7 processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage
- Ultra-low power consumption with integrated sleep modes

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

24.2 Block Diagram

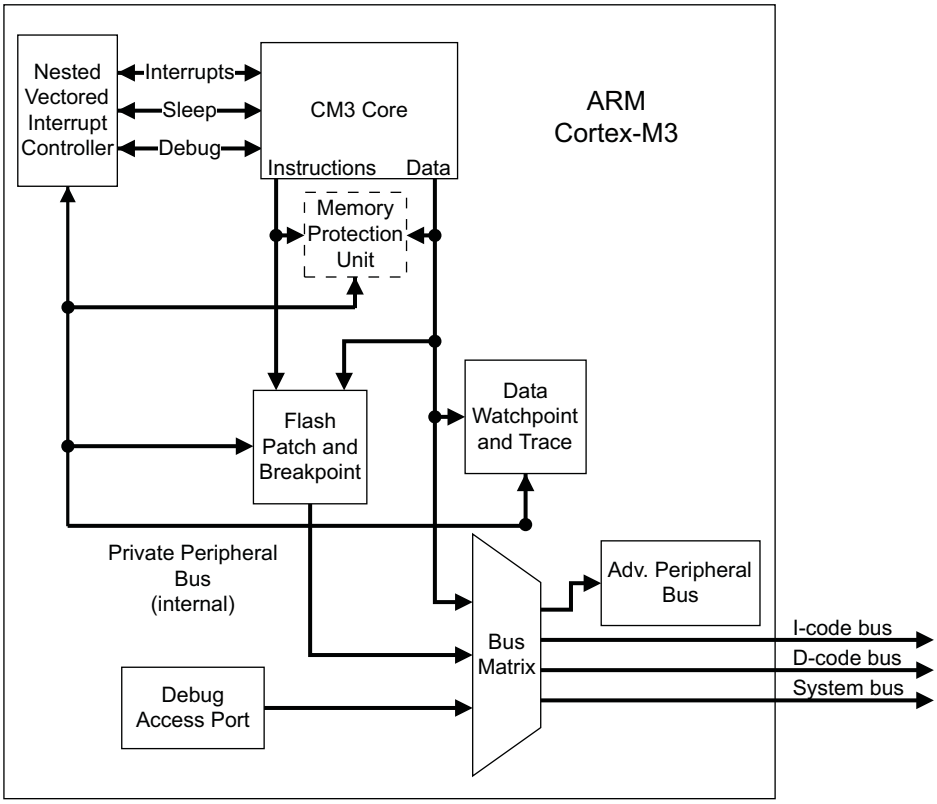
The Cortex-M3 processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. It delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including single-cycle 32x32 multiplication and dedicated hardware division.

To facilitate the design of cost-sensitive devices, this processor implements tightly coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. It implements a version of the Thumb® instruction set, ensuring high code density and reduced program memory requirements. The Cortex-M3 instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

This processor closely integrates a nested interrupt controller (NVIC), to deliver industry-leading interrupt performance. The Concerto™ 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. The block diagram is illustrated below.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 24-1. Cortex-M3 Processor Block Diagram



24.3 Overview

24.3.1 System-Level Interface

The Cortex-M3 processor provides multiple interfaces using AMBA® technology to provide high-speed, low-latency memory accesses. The core supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks, and thread-safe Boolean data handling.

The 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.

24.3.2 System Component Details

The Cortex™-M3 processor 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) in the *Cortex-M3 Peripherals* chapter).
- Nested Vectored Interrupt Controller (NVIC)
  An embedded interrupt controller that supports low latency interrupt processing (see Nested Vectored Interrupt Controller (NVIC) in the *Cortex-M3 Peripherals* chapter).
- 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) in the *Cortex-M3 Peripherals* chapter).
- Memory Protection Unit (MPU)
  Improves system reliability by defining the memory attributes for different memory regions. The MPU

Copyright © 2012–2019, Texas Instruments Incorporated

provides up to eight different regions and an optional predefined background region (see Memory Protection Unit (MPU) in the *Cortex-M3 Peripherals* chapter).

24.4 Programming Model

This section describes the Cortex-M3 programming model. In addition to the individual core register descriptions, information about the processor modes and privilege levels for software execution and stacks is included.

24.4.1 Processor Mode and Privilege Levels for Software Execution

The Cortex-M3 processor 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 processor 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 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.

24.4.2 Stacks

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

In thread mode, the CONTROL register 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 24-1.

Table 24-1. Summary of Processor Mode, Privilege Level, and Stack Use

Processor Mode	Use	Privilege Level	Stack Used
Thread	Applications	Privileged or unprivilegeda	Main stack or process stacka
Handler	Exception handlers	Always privileged	Main stack

24.4.3 Register Map

Figure 24-2 shows the Cortex-M3 register set. Table 24-2 lists the core registers. The core registers are not memory mapped and are accessed by register name, so the base address is not applicable and there is no offset.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 24-2. Cortex-M3 Register Set

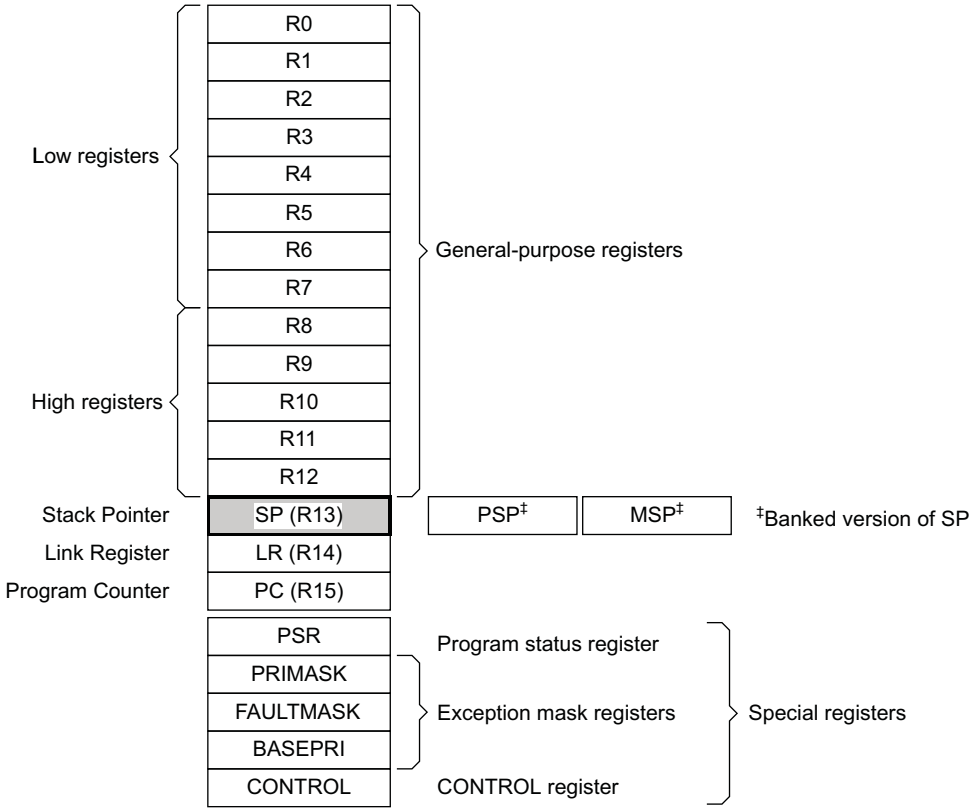


Table 24-2. Processor Register Map

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

Copyright © 2012–2019, Texas Instruments Incorporated

24.4.4 Register Descriptions

This section lists and describes the Cortex-M3 registers, in the order shown in Figure 24-2. 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.

24.4.4.1 Cortex General-Purpose Registers 0-12 (Core R0-R12)

The Core registers (R*n*) are 32-bit general-purpose registers for data operations and can be accessed from either privileged or unprivileged mode.

Figure 24-3. Cortex General-Purpose Registers 0-12 (R0-R12)

31	0
DATA	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-3. Cortex General-Purpose Registers 0-12 (R0-R12) Field Descriptions

Bit	Field	Value	Description
31-0	DATA		Register data.

24.4.4.2 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. 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.

Figure 24-4. Stack Pointer Register (SP)

31	0
SP	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-4. Cortex General-Purpose Registers 0-12 (R0-R12) Field Descriptions

Bit	Field	Value	Description
31-0	SP		Stack pointer address

Copyright © 2012–2019, Texas Instruments Incorporated

24.4.4.3 Link Register (LR)

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

EXC_RETURN is loaded into LR on exception entry. See Table 24-18 for the values and description.

Figure 24-5. Link Register

31	0
LINK	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-5. Link Register Field Descriptions

Bit	Field	Value	Description
31-0	LINK		Return address

24.4.4.4 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.

Figure 24-6. Program Counter Register

31	0
PC	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-6. Program Counter Register Field Descriptions

Bit	Field	Value	Description
31-0	PC		Current program address

24.4.4.5 Program Status Register (PSR)

The program status register (PSR) has three functions, and the register bits are assigned to the different functions:

- Application Program Status Register (APSR), bits 31:27,
- Execution Program Status Register (EPSR), bits 26:24, 15:10
- Interrupt Program Status Register (IPSR), bits 6:0

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

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

EPSR contains the Thumb state bit and the execution state bits for the If-Then (IT) instruction or the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction. Attempts to read the EPSR directly through application software using the MSR instruction always return zero. Attempts to write the EPSR using the MSR instruction in application software are always ignored. Fault handlers can examine the EPSR value in the stacked PSR to determine the operation that faulted (see Section 24.7.7).

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

Copyright © 2012–2019, Texas Instruments Incorporated

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. Table 24-7 shows the possible register combinations for the PSR. See the MRS and MSR instruction descriptions in the *Cortex-M3 Instruction Set Technical User's Manual*for more information about how to access the program status registers.

Table 24-7. PSR Register Combinations

Register	Type	Combination
PSR	R/W(1)(2)	APSR, EPSR, and IPSR
IEPSR	RO	EPSR and IPSR
IAPSR	R/W	APSR and IPSR
EAPSR	R/W	APSR and EPSR

(1) The processor ignores writes to the IPSR bits.
(2) Reads of the EPSR bits return zero, and the processor ignores writes to these bits.

Figure 24-7. Program Status Register (PSR)

31	30	29	28	27	26	25	24
N	Z	C	V	Q	ICI / IT		THUMB
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/0-0x0		R/0-1

23	16
Reserved	
R-0	

15	10	9	8
ICI/IT		Reserved	
R/O-0x00		R-0	

7	6	0
Reserved	ISRNUM	
R-0	R/0-0x00	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-8. Program Status Register (PSR) Field Descriptions

Bit	Field	Value	Description
31	N		APSR Negative or Less Flag
		0	The previous operation result was positive, zero, greater than, or equal
		1	The previous operation result was negative or less than.
30	Z		APSR Zero Flag
		0	The previous operation result was non-zero.
		1	The previous operation result was zero
29	C		APSR Carry or Borrow Flag
		0	The previous add operation did not result in a carry bit or the previous subtract operation resulted in a borrow bit.
		1	The previous add operation resulted in a carry bit or the previous subtract operation did not result in a borrow bit.
			The value of this bit is only meaningful when accessing PSR or APSR.
28	V		APSR Overflow Flag
		0	The previous operation did not result in an overflow
		1	The previous operation resulted in an overflow.
			The value of this bit is only meaningful when accessing PSR or APSR.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 24-8. Program Status Register (PSR) Field Descriptions (continued)

Bit	Field	Value	Description
27	Q		APSR DSP Overflow and Saturation Flag
		0	DSP overflow or saturation has not occurred since reset or since the bit was last cleared.
		1	DSP Overflow or saturation has occurred. DSP overflow or saturation has not occurred since The value of this bit is only meaningful when accessing PSR or APSR. This bit is cleared by software using an MRS instruction.
26-25	ICI / IT		EPSR ICI / IT status 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 a 16-bit IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the Cortex-M3 Instruction Set Technical User's Manual for more information. The value of this field is only meaningful when accessing PSR or EPSR.
24	THUMB	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 Attempting to execute instructions when this bit is clear results in a fault or lockup. See “Lockup” on page 116 for more information. The value of this bit is only meaningful when accessing PSR or EPSR.
23-16	Reserved		Reserved
15-10	ICI / IT		EPSR ICI / IT status These bits, along with bits 26:25, contain the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction or the execution state bits of the IT instruction. When an interrupt occurs during the execution of an LDM, STM, PUSH or POP instruction, the processor stops the load multiple or store multiple instruction operation temporarily and stores the next register operand in the multiple operation to bits 15:12. After servicing the interrupt, the processor returns to the register pointed to by bits 15:12 and resumes execution of the multiple load or store instruction. When EPSR holds the ICI execution state, bits 11:10 are zero. The If-Then block contains up to four instructions following a 16-bit IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the Cortex™-M3 Instruction Set Technical User's Manual for more information. The value of this field is only meaningful when accessing PSR or EPSR.
9-7	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 24-8. Program Status Register (PSR) Field Descriptions (continued)

Bit	Field	Value	Description
6-0	ISRNUM		IPSR ISR Number. This field contains the exception type number of the current Interrupt Service Routine (ISR).
		0h	Thread mode
		1h	Reserved
		2h	NMI
		3h	Hard fault
		4h	Memory management fault
		5h	Bus fault
		6h	Usage fault
		0x07h-0x0Ah	Reserved
		Bh	SVCall
		Ch	Reserved for Debug
		Dh	Reserved
		Eh	PendSV
		Fh	SysTick
		10h	Interrupt Vector 0
		11h	Interrupt Vector 1
		...	...
		6Ah	Interrupt Vector 90
		0x6B-0x7F	Reserved
			See Section 24.7.2 for more information. The value of this field is only meaningful when accessing PSR or IPSR.

24.4.4.6 Priority Mask Register (PRIMASK)

The PRIMASK register prevents activation of all exceptions with programmable priority. Reset, non-maskable interrupt (NMI), and hard fault are the only exceptions with fixed priority. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the PRIMASK register, and the CPS instruction may be used to change the value of the PRIMASK register. See the *Cortex-M3 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see Section 24.7.2.

Figure 24-8. Priority Mask Register (PRIMASK)

31	1	0
Reserved		PRIMASK
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-9. Priority Mask Register (PRIMASK) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	PRIMASK		Priority mask
		0	No effect
		1	Prevents the activation of all exceptions with configurable priority.

Copyright © 2012–2019, Texas Instruments Incorporated

24.4.4.7 Fault Mask Register (FAULTMASK)

The FAULTMASK register prevents activation of all exceptions except for the Non-Maskable Interrupt (NMI). Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the FAULTMASK register, and the CPS instruction may be used to change the value of the FAULTMASK register. See the *Cortex-M3 Instruction Set Technical User's Manual*for more information on these instructions. For more information on exception priority levels, see .

Figure 24-9. Fault Mask Register (FAULTMASK)

31	1	0
Reserved		FAULTMASK
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-10. Fault Mask Register (FAULTMASK) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
0	FAULTMASK		Priority mask
		0	No effect
		1	Prevents the activation of all exceptions except for NMI.

24.4.4.8 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 Section 24.7.2.

Figure 24-10. Base Priority Mask Register (BASEPRI)

31	8	7	5	4	0
Reserved		BASEPRI		Reserved	
R-0		R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-11. Base Priority Mask Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reserved
7-5	BASEPRI		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.
		0h	All exceptions are unmasked.
		1h	All exceptions with priority level 1-7 are masked
		2h	All exceptions with priority level 2-7 are masked
		3h	All exceptions with priority level 3-7 are masked.
		4h	All exceptions with priority level 4-7 are masked.
		5h	All exceptions with priority level 5-7 are masked.
		6h	All exceptions with priority level 6-7 are masked
		7h	All exceptions with priority level 7 are masked.
4-0	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.

Copyright © 2012–2019, Texas Instruments Incorporated

24.4.4.9 Control Register (CONTROL)

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

Handler mode always uses MSP, so the processor ignores explicit writes to the ASP bit of the CONTROL register when in Handler mode. The exception entry and return mechanisms automatically update the CONTROL register based on the EXC_RETURN value (see Table 24-18). In an OS environment, threads running in Thread mode should use the process stack and the kernel and exception handlers should use the main stack. By default, Thread mode uses MSP. To switch the stack pointer used in Thread mode to PSP, either use the MSR instruction to set the ASP bit, as detailed in the *Cortex-M3 Instruction Set Technical User's Manual*, or perform an exception return to thread mode with the appropriate EXC_RETURN value, as shown in Table 24-18.

Note: When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction, ensuring that instructions after the ISB execute use the new stack pointer. See the *Cortex-M3 Instruction Set Technical User's Manual*.

Figure 24-11. Control Register (CONTROL)

31	1	0
Reserved	ASP	TMPL
R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24-12. Control Register (CONTROL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved		Reserved
0	ASP		Active Stack Pointer
		0	MSP is the current stack pointer
		1	PSP is the current stack pointer.
			In Handler mode, this bit reads as zero and ignores writes. The processor updates this bit automatically on exception return.
	TMPL		Thread mode privilege level
		0	Only privileged software can be executed in thread mode.
		1	Unprivileged software can be executed in thread mode.

24.4.5 Exceptions and Interrupts

The 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 Section 24.7.7 for more information.

The NVIC registers control interrupt handling. See Nested Vectored Interrupt Controller (NVIC) in the *Cortex-M3 Peripherals* chapter for more information.

24.4.6 Data Types

The processor supports 32-bit words, 16-bit halfwords, and 8-bit bytes. It also supports 64-bit data transfer instructions. All instruction and data memory accesses are little endian. See Section 24.6 for more information.

24.5 Memory Model

This section describes the behavior of memory accesses and the bit-banding features. The memory map for the controller is provided in the data manual.

The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data (see Section 24.6.4).

Copyright © 2012–2019, Texas Instruments Incorporated

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers (see *the Cortex-M3 Peripherals* chapter).

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

24.6 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.

24.6.1 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 Section 24.6.3).

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.

24.6.2 Behavior of Memory Accesses

Table 24-13 shows the behavior of accesses to each region in the memory map. See Section 24.6 for more information on memory types and the XN attribute. Concerto™ devices may have reserved memory areas within the address ranges shown below.

Table 24-13. 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. Includes Boot ROM and Flash on Concerto devices.
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 24-14).
0x4000.0000 - 0x5FFF.FFFF	Peripheral	Device	XN	This region includes bit band and bit band alias areas (see Table 24-15).
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	-	-	-

Copyright © 2012–2019, Texas Instruments Incorporated

The Code, SRAM, and external RAM regions can hold programs. However, it is recommended that programs always use the Code region because the Cortex-M3 has separate buses that can perform instruction fetches and data accesses simultaneously.

The MPU can override the default memory access behavior described in this section. For more information, see the Memory Protection Unit (MPU) section in the *Cortex-M3 Peripherals* chapter.

Cortex-M3 prefetches instructions ahead of execution and speculatively prefetches from branch target addresses.

24.6.3 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.

Section 24.6.1 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 processor 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 codeIf 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 switchingIf 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 changeWhen 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 *Cortexâ„¢-M3 Instruction Set Technical User's Manual*.

Copyright © 2012–2019, Texas Instruments Incorporated

24.6.4 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 24-14. Accesses to the 32-MB peripheral alias region map to the 1-MB peripheral bit-band region, as shown in Table 24-15.

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 24-14. SRAM Memory Bit-Banding Regions

Address Range	Memory Region	Instruction and Data Accesses
0x2000.0000 - 0x200F.FFFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias.
0x2200.0000 - 0x23FF.FFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

Table 24-15. Peripheral Memory Bit-Banding Regions

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

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

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

bit_word_addr = bit_band_base + bit_word_offset

where:

bit_word_offset

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

bit_word_addr

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

bit_band_base

The starting address of the alias region.

byte_offset

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

bit_number

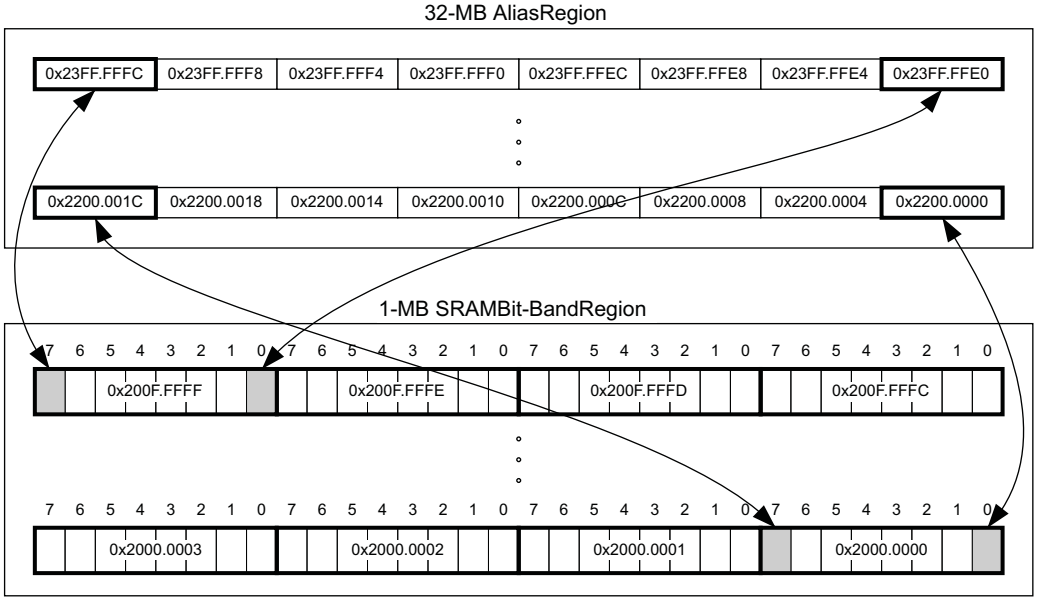
The bit position, 0-7, of the targeted bit.

Figure 24-12 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

Copyright © 2012–2019, Texas Instruments Incorporated

- 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 24-12. Bit-Band Mapping



24.6.4.1 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.

24.6.4.2 Directly Accessing a Bit-Band Region

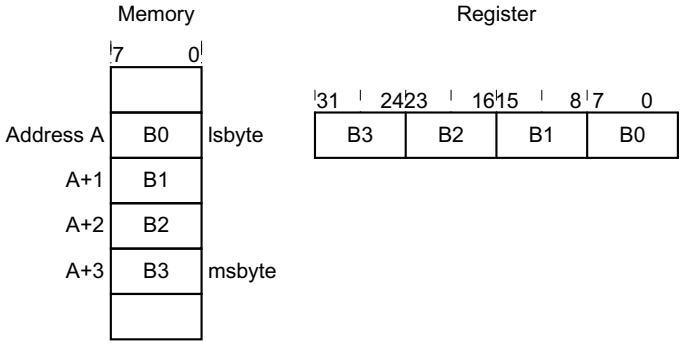
Section 24.6.2 describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

24.6.5 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 24-13 illustrates how data is stored.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 24-13. Data Storage



24.6.6 Synchronization Primitives

The Cortex-M3 instruction set includes pairs of synchronization primitives which provide a non-blocking mechanism that a thread or process can use to obtain exclusive access to a memory location. Software can use these primitives to perform a guaranteed read-modify-write memory update sequence or for a semaphore mechanism.

A pair of synchronization primitives consists of:

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

The pairs of load-exclusive and store-exclusive instructions are:

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

Software must use a load-exclusive instruction with the corresponding store-exclusive instruction.

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

- Use a load-exclusive instruction to read the value of the location.
- Update the value, as required.
- Use a store-exclusive instruction to attempt to write the new value back to the memory location, and test the returned status bit. If the status bit is clear, the read-modify-write completed successfully; if the status bit is set, no write was performed, which indicates that the value returned at step 1 might be out of date. The software must retry the read-modify-write sequence.

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

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

The Cortex-M3 processor 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

Copyright © 2012–2019, Texas Instruments Incorporated

threads.

For more information about the synchronization primitive instructions, see the *Cortex-M3 Instruction Set Technical User's Manual*.

24.7 Exception Model

The ARM Cortex-M3 processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration.

Table 24-16 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 55 interrupts (listed in Table 24-17).

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) in the *Cortex-M3 Peripherals* chapter.

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.

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

See Nested Vectored Interrupt Controller (NVIC) in the *Cortex-M3 Peripherals* chapter for more information on exceptions and interrupts.

24.7.1 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.NoteAn 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.

24.7.2 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

Copyright © 2012–2019, Texas Instruments Incorporated

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. On Concerto devices, a clock fail condition, C28 NMI watchdog timeout reset, C28 PIE NMI vector fetch error, and GPIO NMI input trigger, can trigger an NMI condition.

- **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. On Concerto devices, RAM uncorrectable errors (address error, parity error, and double data error), and flash uncorrectable errors, can trigger a bus default.
- **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 24-17 lists the interrupts on the 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 24-16 shows as having configurable priority (see the SYSHNDCTRL register and the DIS0 register in the *Cortex-M3 Peripherals* chapter.

For more information about hard faults, memory management faults, bus faults, and usage faults, see Section 24.8.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 24-16. Exception Types Description

Exception Type	Vector Number	Priority[1]	Vector Address or Offset[2]	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 On Concerto devices activated by clock fail condition, C28 PIE error, external M3GPIO NMI input signal, and C28 NMI WD timeout reset.
Hard Fault	3	-1	0x0000.000C	-
Memory Management	4	programmable[3]	0x0000.0010	Synchronous
Bus Fault	5	programmable[3]	0x0000.0014	Synchronous when precise and asynchronous when imprecise. On Concerto devices activated by memory access errors and RAM and flash uncorrectable data errors.
Usage Fault	6	programmable[3]	0x0000.0018	Synchronous
-	7-10	-	-	Reserved
SVCall	11	programmable[3]	0x0000.002C	Synchronous
Debug Monitor	12	programmable[3]	0x0000.0030	Synchronous
-	13	-	-	Reserved
PendSV	14	programmablec[3]	0x0000.0038	Asynchronous
SysTick	15	programmable[3]	0x0000.003C	Asynchronous
Interrupts	16 and above	programmable [4]	0x0000.0040 and above	Asynchronous

(1) 0 is the default priority for all the programmable priorities
(2) See Section 24.7.4
(3) See SYSPRI1 in the *Cortex-M3 Peripherals* chapter.
(4) See PRIn registers in the *Cortex-M3 Peripherals* chapter

Table 24-17. 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	I2C0
25-33	9-17	-	Reserved
34	18	0x0000.0088	Watchdog Timers 0 and 1
35	19	0x0000.008C	Timer 0A
36	20	0x0000.0090	Timer 0B

Copyright © 2012–2019, Texas Instruments Incorporated

Table 24-17. Interrupts (continued)

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
37	21	0x0000.0094	Timer 1A
38	22	0x0000.0098	Timer 1B
39	23	0x0000.009C	Timer 2A
40	24	0x0000.00A0	Timer 2B
41-43	25-27	-	Reserved
44	28	0x0000.00B0	System Control
45	29	0x0000.00B4	Reserved
46	30	0x0000.00B8	GPIO Port F
47	31	0x0000.00BC	GPIO Port G
48	32	0x0000.00C0	GPIO Port H
49	33	0x0000.00C4	UART2
50	34	0x0000.00C8	SSI1
51	35	0x0000.00CC	Timer 3A
52	36	0x0000.00D0	Timer 3B
53	37	0x0000.00D4	I2C1
54-57	38-41	-	Reserved
58	42	0x0000.00E8	Ethernet Controller
60	44	0x0000.00F0	USB
61	45	-	Reserved
62	46	0x0000.00F8	µDMA Software
63	47	0x0000.00FC	µDMA Error
64-68	48-52	-	Reserved
69	53	0x0000.0114	EPI
70	54	0x0000.0118	GPIO Port J
71-72	55-56	-	Reserved
73	57	0x0000.0124	SSI 2
74	58	0x0000.0128	SSI 3
75	59	0x0000.012C	UART3
76	60	0x0000.0130	UART4
77-79	61-63	-	Reserved
80	64	0x0000.0140	CAN0 INT0
81	65	0x0000.0144	CAN0 INT1
82	66	0x0000.0148	CAN1 INT0
83	67	0x0000.014C	CAN1 INT1
84-87	68-71	-	Reserved
88	72	0x0000.0160	ADCINT1
89	73	0x0000.0164	ADCINT2
90	74	0x0000.0168	ADCINT3
91	75	0x0000.016C	ADCINT4
92	76	0x0000.0170	ADCINT5
93	77	0x0000.0174	ADCINT6
94	78	0x0000.0178	ADCINT7
95	79	0x0000.017C	ADCINT8
96	80	0x0000.0180	CTOMIPC1
97	81	0x0000.0184	CTOMIPC2
98	82	0x0000.0188	CTOMIPC3
99	83	0x0000.018C	CTOMIPC4

Copyright © 2012–2019, Texas Instruments Incorporated

Table 24-17. Interrupts (continued)

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
100-103	84-87	-	Reserved
104	88	0x0000.01A0	RAM Single Error
1-5	89	0x0000.01A4	System / USB PLL Out of Lock
106	90	0x0000.01A8	M3 Flash Single Error
107	91	0x0000.01AC	Reserved

24.7.3 Exception Handlers

The processor handles exceptions using:

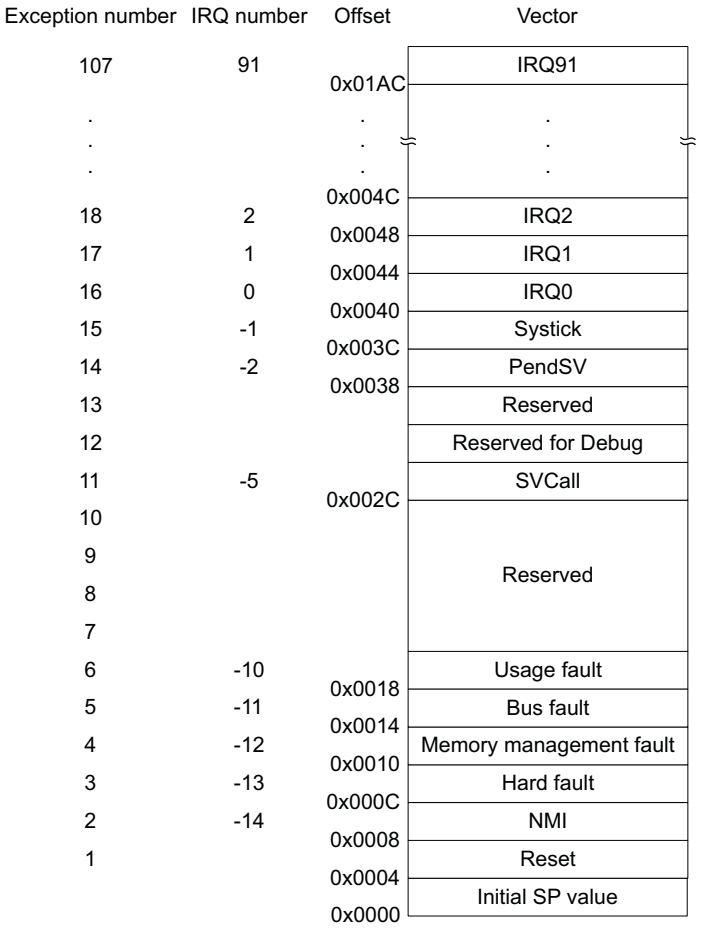
- **Interrupt Service Routines (ISRs).** Interrupts (IRQx) are the exceptions handled by ISRs.
- **Fault Handlers.** Hard fault, memory management fault, usage fault, and bus fault are fault exceptions handled by the fault handlers.
- **System Handlers.** NMI, PendSV, SVCall, SysTick, and the fault exceptions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

24.7.4 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 24-16. Figure 24-14 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.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 24-14. Vector table



On system reset, the vector table is fixed at address 0x0000.0000. Privileged software can write to the Vector Table Offset (VTABLE) register to relocate the vector table start address to a different memory location, in the range 0x0000.0200 to 0x3FFF.FE00. Note that when configuring the VTABLE register, the offset must be aligned on a 512-byte boundary.

24.7.5 Exception Priorities

As Table 24-16 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 the *Cortex-M3 Peripherals* chapter.

NOTE: Configurable priority values for the Concerto 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].

Copyright © 2012–2019, Texas Instruments Incorporated

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.

24.7.6 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 Application Interrupt and Reset Control (APINT) section in the the *Cortex-M3 Peripherals* chapter.

24.7.7 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 Section 24.7.6 for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See Section 24.7.7.1 for 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 Section 24.7.7.2 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.

24.7.7.1 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.

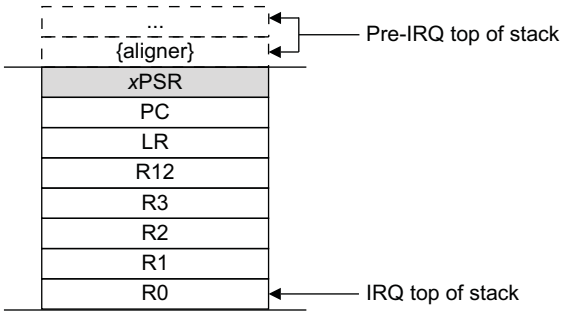
When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has more priority than any limits set by the mask registers (see the Priority Mask Register (PRIMASK), Fault Mask Register (FAULTMASK), and Base Priority Mask Register (BASEPRI). 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*.

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 24-15. Exception Stack Frame



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

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

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

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

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

24.7.7.2 Exception Return

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

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

EXC_RETURN is the value loaded into the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest four bits of this value provide information on the return stack and processor mode. Table 24-18 shows the EXC_RETURN values with a description of the exception return behavior.

EXC_RETURN bits 31:4 are all set. When this value is loaded into the PC, it indicates to the processor that the exception is complete, and the processor initiates the appropriate exception return sequence.

Table 24-18. Exception Return Behavior

EXC_RETURN[31:0]	Description
0xFFFF.FFF0	Reserved
0xFFFF.FFF1	Return to Handler mode.Exception return uses state from MSP.Execution uses MSP after return.
0xFFFF.FFF2 - 0xFFFF.FFF8	Reserved
0xFFFF.FFF9	Return to Thread mode.Exception return uses state from MSP.Execution uses MSP after return.
0xFFFF.FFFA - 0xFFFF.FFFC	Reserved
0xFFFF.FFFD	Return to Thread mode.Exception return uses state from PSP.Execution uses PSP after return.
0xFFFF.FFFE - 0xFFFF.FFFF	Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

24.8 Fault Handling

Faults are a subset of the exceptions (see Section 24.7.)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.

24.8.1 Fault Types

Table 24-19 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 the *Cortex-M3 Peripherals* chapter for more information about the fault status registers.

Table 24-19. 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(1)
MPU or default memory mismatch on data access	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	DERR
MPU or default memory mismatch on exception stacking	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	MSTKE
MPU or default memory mismatch on exception unstacking	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	MUSTKE
Bus error during exception stacking	Bus fault	Bus Fault Status (BFAULTSTAT)	BSTKE
Bus error during exception unstacking	Bus fault	Bus Fault Status (BFAULTSTAT)	BUSTKE
Bus error during instruction prefetch	Bus fault	Bus Fault Status (BFAULTSTAT)	IBUS
Precise data bus error	Bus fault	Bus Fault Status (BFAULTSTAT)	PRECISE
Imprecise data bus error	Bus fault	Bus Fault Status (BFAULTSTAT)	IMPRE
Attempt to access a coprocessor	Usage fault	Usage Fault Status (UFAULTSTAT)	NOCP
Undefined instruction	Usage fault	Usage Fault Status (UFAULTSTAT)	UNDEF
Attempt to enter an invalid instruction set state(2)	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

(1) Occurs on an access to an XN region even if the MPU is disabled.

(2) Attempting to use an instruction set other than the Thumb instruction set, or returning to a non load-store-multiple instruction with ICI continuation

Copyright © 2012–2019, Texas Instruments Incorporated

24.8.2 Fault Escalation and Hard Faults

All fault exceptions except for hard fault have configurable exception priority (see SYSPRI1 in the *Cortex-M3 Peripherals* chapter. Software can disable execution of the handlers for these faults (see SYSHNDCTRL in the *Cortex-M3 Peripherals* chapter.

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 Section 24.7.

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.

24.8.3 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 24-20.

Table 24-20. Fault Status and Fault Address Registers

Handler	Status Register Name	Address Register Name
Hard fault	Hard Fault Status (HFAULTSTAT)	-
Memory management fault	Memory Management Fault Status (MFAULTSTAT)	Memory Management Fault Address (MMADDR)
Bus fault	Bus Fault Status (BFAULTSTAT)	Bus Fault Address (FAULTADDR)
Usage fault	Usage Fault Status (UFAULTSTAT)	-

24.8.4 Lockup

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

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

24.9 Power Management

The Cortex-M3 processor sleep mode and deep-sleep mode reduces power consumptions:

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

Copyright © 2012–2019, Texas Instruments Incorporated

The SLEEPDEEP bit of the System Control (SYSCTRL) register selects which sleep mode is used (see the *Cortex-M3 Peripherals* chapter). For more information about the behavior of the sleep modes, see the *System Control* chapter.

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.

24.9.1 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.

24.9.1.1 Wait for Interrupt

The wait for interrupt instruction, WFI, causes immediate entry to sleep mode unless the wake-up condition is true (see Section 24.9.2.1). When the processor executes a WFI instruction, it stops executing instructions and enters sleep mode. See the *Cortex-M3 Instruction Set Technical User's Manual*for more information.

24.9.1.2 Wait for Event

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

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

See the *Cortex-M3 Instruction Set Technical User's Manual* for more information.

24.9.1.3 Sleep-on-Exit

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

24.9.2 Wake Up from Sleep Mode

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

24.9.2.1 Wake Up from WFI or Sleep-on-Exit

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

24.9.2.2 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 the *Cortex-M3 Peripherals* chapter.

Copyright © 2012–2019, Texas Instruments Incorporated

24.10 Instruction Set Summary

The processor implements a version of the Thumb instruction set. Table 24-21 lists the supported instructions.

Notes:

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

For more information on the instructions and operands, see the instruction descriptions in the *Cortex-M3 Instruction Set Technical User's Manual*.

Table 24-21. Cortex-M3 Instruction Summary

Mnemonic	Operands	Brief Description	Flags
ADC, ADCS	{Rd,} Rn , Op2	Add with carry	N,Z,C,V
ADD, ADDS	{Rd,} Rn , Op2	Add	N,Z,C,V
ADD, ADDW	{Rd,} Rn , #imm12	Add	N,Z,C,V
ADR	Rd , label	Load PC-relative address	-
AND, ANDS	{Rd ,} Rn , Op2	Logical AND	N,Z,C
ASR, ASRS	Rd , Rm , <Rs|#n>	Arithmetic shift right	N,Z,C
B	label	Branch	-
BFC	Rd , #lsb , #width	Bit field clear	-
BFI	Rd , Rn , #lsb , #width	Bit field insert	-
BIC, BICS	{Rd ,} Rn , Op2	Bit clear	N,Z,C
BKPT	#imm	Breakpoint	-
BL	label	Branch with link	-
BLX	Rm	Branch indirect with link	-
BX	Rm	Branch indirect	-
CBNZ	Rn , label	Compare and branch if non-zero	-
CBZ	Rn , label	Compare and branch if zero	-
CLREX	-	Clear exclusive	-
CLZ	Rd , Rm	Count leading zeros	-
CMN	Rn , Op2	Compare negative	N,Z,C,V
CMP	Rn , Op2	Compare	N,Z,C,V
CPSID	iflags	Change processor state, disable interrupts	-
CPSIE	iflags	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	-	Ifâ€˜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	-

Copyright © 2012–2019, Texas Instruments Incorporated

Table 24-21. Cortex-M3 Instruction Summary (continued)

Mnemonic	Operands	Brief Description	Flags
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 words	-
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 , Rn	Move from general register to special register	N,Z,C,V
MUL, MULS	{Rd,}Rn , Rm	Multiply, 32-bit result	N,Z
MVN, MVNS	Rd , Op2	Move NOT	N,Z,C
NOP	-	No operation	-
ORN, ORNS	{Rd,} Rn , Op2	Logical OR NOT	N,Z,C
ORR, ORRS	{Rd,} Rn , Op2	Logical OR	N,Z,C
POP	reglist	Pop registers from stack	-
PUSH	reglist	Push registers onto stack	-
RBIT	Rd , Rn	Reverse bits	-
REV	Rd , Rn	Reverse byte order in a word	-
REV16	Rd , Rn	Reverse byte order in each halfword	-
REVSH	Rd , Rn	Reverse byte order in bottom halfword and sign extend	-
ROR, RORS	Rd , Rm , <Rs|#n>	Rotate right	N,Z,C
RRX, RRXS	Rd , Rm	Rotate right with extend	N,Z,C
RSB, RSBS	{Rd,} Rn , Op2	Reverse subtract	N,Z,C,V
SBC, SBCS	{Rd,} Rn , Op2	Subtract with carry	N,Z,C,V
SBFX	Rd , Rn , #lsb , #width	Signed bit field extract	-
SDIV	{Rd ,} Rn , Rm	Signed divide	-
SEV	-	Send event	-
SMLAL	RdLo, RdHi, Rn, Rm	Signed multiply with accumulate (32x32+64), 64-bit result	-
SMULL	RdLo, RdHi, Rn, Rm	Signed multiply (32x32), 64-bit result	-

Copyright © 2012–2019, Texas Instruments Incorporated

Table 24-21. Cortex-M3 Instruction Summary (continued)

Mnemonic	Operands	Brief Description	Flags
SSAT	Rd, #n, Rm {,shift #s}	Signed saturate	Q
STM	Rn{!} , reglist	Store multiple registers, increment after	-
STMDB, STMEA	Rn{!} , reglist	Store multiple registers, decrement before	-
STMFD, STMIA	Rn{!} , reglist	Store multiple registers, increment after	-
STR	Rt , [Rn {, #offset}]	Store register word	-
STRB, STRBT	Rt , [Rn {, #offset}]	Store register byte	-
STRD	Rt , Rt2 , [Rn {, #offset}]	Store register two words	-
STREX	Rd , Rt , [Rn , #offset]	Store register exclusive	-
STREXB	Rd , Rt , [Rn]	Store register exclusive byte	-
STREXH	Rd , Rt , [Rn]	Store register exclusive halfword	-
STRH, STRHT	Rt , [Rn {, #offset}]	Store register halfword	-
STRSB, STRSBT	Rt , [Rn {, #offset}]	Store register signed byte	-
STRSH, STRSHT	Rt , [Rn {, #offset}]	Store register signed halfword	-
STRT	Rt , [Rn {, #offset}]	Store register word	-
SUB, SUBS	{Rd,} Rn , Op2	Subtract	N,Z,C,V
SUB, SUBW	{Rd,} Rn , #imm12	Subtract 12-bit constant	N,Z,C,V
SVC	#imm	Supervisor call	-
SXTB	{Rd,} Rm {,ROR #n}	Sign extend a byte	-
SXTH	{Rd,} Rm {,ROR #n}	Sign extend a halfword	-
TBB	[Rn, Rm]	Table branch byte	-
TBH	[Rn, Rm, LSL #1]	Table branch halfword	-
TEQ	Rn, Op2	Test equivalence	N,Z,C
TST	Rn, Op2	Test	N,Z,C
UBFX	Rd , Rn , #lsb , #width	Unsigned bit field extract	-
UDIV	{Rd,} Rn , Rm	Unsigned divide	-
UMLAL	RdLo, RdHi, Rn, Rm	Unsigned multiply with accumulate (32x32+32+32), 64-bit result	-
UMULL	RdLo, RdHi, Rn, Rm	Unsigned multiply (32x 2), 64-bit result	-
USAT	Rd, #n, Rm {,shift #s}	Unsigned saturate	Q
UXTB	{Rd,} Rm {,ROR #n}	Zero extend a byte	-
UXTH	{Rd,} Rm {,ROR #n}	Zero extend a halfword	-
WFE	-	Wait for event	-
WFI	-	Wait for interrupt	-

Copyright © 2012–2019, Texas Instruments Incorporated

Cortex-M3 Peripherals

This chapter provides information on the Cortex-M3 processor peripherals.

Topic **Page**

25.1 Overview **1599**
25.2 Functional Description **1599**
25.3 Register Map **1606**
25.4 System Timer (SysTick) Register Descriptions **1609**
25.5 NVIC Register Descriptions **1611**
25.6 System Control Block (SCB) Register Descriptions **1628**
25.7 Memory Protection Unit (MPU) Register Descriptions **1649**

Copyright © 2012–2019, Texas Instruments Incorporated

25.1 Overview

The Concerto™ implementation of the Cortex-M3 processor peripherals include:

- SysTick
  Provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism.
- Nested Vectored Interrupt Controller (NVIC)
  - Facilitates low-latency exception and interrupt handling
  - Controls power management
  - Implements system control registers
- System Control Block (SCB)
  Provides system implementation information and system control, including configuration, control, and reporting of system exceptions.
- Memory Protection Unit (MPU)
  Supports the standard ARMv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.
  Note: This feature is disabled on these devices.

Table 25-1 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 25-1. Core Peripheral Register Regions

Address	Core Peripheral
0xE000.E010-0xE000.E01F	System Timer
0xE000.E100-0xE000.E4EF 0xE000.EF00-0xE000.EF03	Nested Vectored Interrupt Controller
0xE000.E008-0xE000.E00F 0xE000.ED00-0xE000.ED3F	System Control Block
0xE000.ED90-0xE000.EDB8	Memory Protection Unit

25.2 Functional Description

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

25.2.1 System Timer (SysTick)

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

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

The timer consists of three registers:

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

Copyright © 2012–2019, Texas Instruments Incorporated

wrap value.

- SysTick Current Value (STCURRENT): The current value of the counter.

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

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

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

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

25.2.2 Nested Vectored Interrupt Controller (NVIC)

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

- 53 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.

25.2.2.1 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 below 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.

25.2.2.2 Hardware and Software Control of Interrupts

The Cortex-M3 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

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

Copyright © 2012–2019, Texas Instruments Incorporated

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.

25.2.3 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.

25.2.4 Memory Protection Unit (MPU)

NOTE: This feature is disabled on these devices.

The MPU divides the memory map into a number of regions and defines the location, size, access permissions, and memory attributes of each region. The MPU supports independent attribute settings for each region, overlapping regions, and export of memory attributes to the system.

The memory attributes affect the behavior of memory accesses to the region. The Cortex-M3 MPU defines eight separate memory regions, 0-7, and a background region.

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

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

The Cortex-M3 MPU memory map is unified, meaning that instruction accesses and data accesses have the same region settings.

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

Configuration of MPU regions is based on memory types. See Memory Regions, Types and Attributes in the *Cortex-M3 Processor* chapter for more information.

Table 25-2 shows the possible MPU region attributes. See Section 25.2.4.2.1 for guidelines for programming a microcontroller implementation.

Table 25-2. Memory Attributes Summary

Memory Type	Description
Strongly Ordered	All accesses to Strongly Ordered memory occur in program order.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-2. Memory Attributes Summary (continued)

Memory Type	Description
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.

25.2.4.1 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.

25.2.4.1.1 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.

Copyright © 2012–2019, Texas Instruments Incorporated

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.

As an 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.

25.2.4.1.2 Updating an MPU Region Using Multi-Word Writes

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

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

An STM instruction can be used to optimize this:

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

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

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

An STM instruction can be used to optimize this:

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

25.2.4.1.3 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 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.

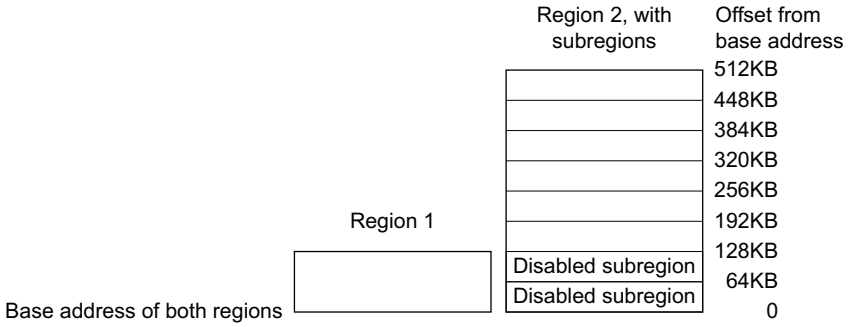
Regions of 32, 64, and 128 bytes do not support subregions. With regions of these sizes, the SRD field must be configured to 0x00, otherwise the MPU behavior is unpredictable.

Copyright © 2012–2019, Texas Instruments Incorporated

25.2.4.1.4 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 25-1 shows.

Figure 25-1. SRD Use Example



25.2.4.2 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 25-3 shows the encodings for the TEX, C, B, and S access permission bits. All encodings are shown for completeness, however the current implementation of the Cortex-M3 does not support the concept of cacheability or shareability. Refer to Section 25.2.4.2.1 for information on programming the MPU for Concerto implementations.

Table 25-3. TEX, S, C, and B Bit Field Encoding

TEX	S	C	B	Memory Type	Shareability	Other Attributes
000b	x[1]	0	0	Strongly Ordered	Shareable	-
000	x[1]	0	1	Device	Shareable	-
000	0	1	0	Normal	Not shareable	Outer and inner write-through. No write allocate.
000	1	1	0	Normal	Shareable	
000	0	1	1	Normal	Not shareable	
000	1	1	1	Normal	Shareable	
001	0	0	0	Normal	Not shareable	Outer and inner non-cacheable.
001	1	0	0	Normal	Shareable	
001	x[1]	0	1	Reserved encoding	-	-
001	x[1]	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[1]	0	0	Device	Not shareable	Non-shared device.
010	x[1]	0	1	Reserved encoding	-	-
010	x[1]	1	x[1]	Reserved encoding	-	-

(1) The MPU ignores the value of this bit.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-3. TEX, S, C, and B Bit Field Encoding (continued)

TEX	S	C	B	Memory Type	Shareability	Other Attributes
1BB	0	A	A	Normal	Not shareable	Cached memory (BB = outer policy, AA = inner policy). See Table 25-4 for the encoding of the AA and BB bits.
1BB	1	A	A	Normal	Shareable	

Table 25-4 shows the cache policy for memory attribute encodings with a TEX value in the range of 0x4-0x7.

Table 25-4. 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 25-5 shows the AP encodings in the MPUATTR register that define the access permissions for privileged and unprivileged software.

Table 25-5. AP Bit Field Encoding

AP Bit Field	Privileged Permissions	Unprivileged Permissions	Description
000	No access	No access	All accesses generate a permission fault.
001	R/W	No access	Access from privileged software only.
010	R/W	RO	Writes by unprivileged software generate a permission fault.
011	R/W	R/W	Full access.
100	Unpredictable	Unpredictable	Reserved.
101	RO	No access	Reads by privileged software only.
110	RO	RO	Read-only, by privileged or unprivileged software.
111	RO	RO	Read-only, by privileged or unprivileged software.

25.2.4.2.1 MPU Configuration for a Concerto Microcontroller

Concerto microcontrollers' MPU should be programmed as shown in Table 25-6.

Table 25-6. Memory Region Attributes for Concerto 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

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-6. Memory Region Attributes for Concerto Microcontrollers (continued)

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

In current Concerto 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.

25.2.4.3 MPU Mismatch

When an access violates the MPU permissions, the processor generates a memory management fault (see the *Cortex-M3 Processor* chapter for more information). The MFAULTSTAT register indicates the cause of the fault.

25.3 Register Map

Table 25-7 lists the Cortex-M3 Peripheral SysTick, NVIC, SCB, and MPU registers. The offset listed is a hexadecimal increment to the register's address, relative to the Core Peripherals base address of 0xE000.E000 (ending address of 0xE000.EFFF).

Note

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

Table 25-7. Peripherals Register Map

Offset	Name	Type	Reset	Description
0x010	STCTRL	R/W	0x0000.0004	SysTick Control and Status Register
0x014	STRELOAD	R/W	0x0000.0000	SysTick Reload Value Register
0x018	STCURRENT	R/WC	0x0000.0000	SysTick Current Value Register
Nested Vectored Interrupt Controller (NVIC) Registers				
0x100	EN0	R/W	0x0000.0000	Interrupt 0-31 Set Enable
0x104	EN1	R/W	0x0000.0000	Interrupt 32-63 Set Enable
0x108	EN2	R/W	0x0000.0000	Interrupt 64-95 Set Enable
0x10C	EN3	R/W	0x0000.0000	Interrupt 96-127 Set Enable
0x110	EN4	R/W	0x0000.0000	Interrupt 128-133 Set Enable
0x180	DIS0	R/W	0x0000.0000	Interrupt 0-31 Clear Enable
0x184	DIS1	R/W	0x0000.0000	Interrupt 32-63 Clear Enable
0x188	DIS2	R/W	0x0000.0000	Interrupt 64-95 Clear Enable
0x18C	DIS3	R/W	0x0000.0000	Interrupt 96-127 Clear Enable
0x190	DIS4	R/W	0x0000.0000	Interrupt 128-133 Clear Enable
0x200	PEND0	R/W	0x0000.0000	Interrupt 0-31 Set Pending
0x204	PEND1	R/W	0x0000.0000	Interrupt 32-63 Set Pending
0x208	PEND2	R/W	0x0000.0000	Interrupt 64-95 Set Pending
0x20C	PEND3	R/W	0x0000.0000	Interrupt 96-127 Set Pending
0x210	PEND4	R/W	0x0000.0000	Interrupt 128-133 Set Pending
0x280	UNPEND0	R/W	0x0000.0000	Interrupt 0-31 Clear Pending
0x284	UNPEND1	R/W	0x0000.0000	Interrupt 32-63 Clear Pending

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-7. Peripherals Register Map (continued)

Offset	Name	Type	Reset	Description
0x288	UNPEND2	R/W	0x0000.0000	Interrupt 64-95 Clear Pending
0x28C	UNPEND3	R/W	0x0000.0000	Interrupt 96-127 Clear Pending
0x290	UNPEND4	R/W	0x0000.0000	Interrupt 128-133 Clear Pending
0x300	ACTIVE0	RO	0x0000.0000	Interrupt 0-31 Active Bit
0x304	ACTIVE1	RO	0x0000.0000	Interrupt 32-63 Active Bit
0x308	ACTIVE2	RO	0x0000.0000	Interrupt 64-95 Active Bit
0x30C	ACTIVE3	RO	0x0000.0000	Interrupt 96-127 Active Bit
0x310	ACTIVE4	RO	0x0000.0000	Interrupt 128-133 Active Bit
0x400	PRI0	R/W	0x0000.0000	Interrupt 0-3 Priority
0x404	PRI1	R/W	0x0000.0000	Interrupt 4-7 Priority
0x408	PRI2	R/W	0x0000.0000	Interrupt 8-11 Priority
0x40C	PRI3	R/W	0x0000.0000	Interrupt 12-15 Priority
0x410	PRI4	R/W	0x0000.0000	Interrupt 16-19 Priority
0x414	PRI5	R/W	0x0000.0000	Interrupt 20-23 Priority
0x418	PRI6	R/W	0x0000.0000	Interrupt 24-27 Priority
0x41C	PRI7	R/W	0x0000.0000	Interrupt 28-31 Priority
0x420	PRI8	R/W	0x0000.0000	Interrupt 32-35 Priority
0x424	PRI9	R/W	0x0000.0000	Interrupt 36-39 Priority
0x428	PRI10	R/W	0x0000.0000	Interrupt 40-43 Priority
0x42C	PRI11	R/W	0x0000.0000	Interrupt 44-47 Priority
0x430	PRI12	R/W	0x0000.0000	Interrupt 48-51 Priority
0x434	PRI13	R/W	0x0000.0000	Interrupt 52-55 Priority
0x438	PRI14	R/W	0x0000.0000	Interrupt 56-59 Priority
0x43C	PRI15	R/W	0x0000.0000	Interrupt 60-63 Priority
0x440	PRI16	R/W	0x0000.0000	Interrupt 64-67 Priority
0x444	PRI17	R/W	0x0000.0000	Interrupt 68-71 Priority
0x448	PRI18	R/W	0x0000.0000	Interrupt 72-75 Priority
0x44C	PRI19	R/W	0x0000.0000	Interrupt 76-79 Priority
0x450	PRI20	R/W	0x0000.0000	Interrupt 80-83 Priority
0x454	PRI21	R/W	0x0000.0000	Interrupt 84-87 Priority
0x458	PRI22	R/W	0x0000.0000	Interrupt 88-91 Priority
0x45C	PRI23	R/W	0x0000.0000	Interrupt 92-95 Priority
0x460	PRI24	R/W	0x0000.0000	Interrupt 96-99 Priority
0x464	PRI25	R/W	0x0000.0000	Interrupt 100-103 Priority
0x468	PRI26	R/W	0x0000.0000	Interrupt 104-107 Priority
0x46C	PRI27	R/W	0x0000.0000	Interrupt 108-111 Priority
0x470	PRI28	R/W	0x0000.0000	Interrupt 112-115 Priority
0x474	PRI29	R/W	0x0000.0000	Interrupt 116-119 Priority
0x478	PRI30	R/W	0x0000.0000	Interrupt 120-123 Priority
0x47C	PRI31	R/W	0x0000.0000	Interrupt 124-127 Priority
0x480	PRI32	R/W	0x0000.0000	Interrupt 128-131 Priority
0x484	PRI33	R/W	0x0000.0000	Interrupt 132-135 Priority
0xF00	SWTRIG	WO	0x0000.0000	Software Trigger Interrupt
System Control Block (SCB) Registers				
0x008	ACTLR	R/W	0x0000.0000	Auxiliary Control
0xD00	CPUID	RO	0x412F.C230	CPU ID Base
0xD04	INTCTRL	R/W	0x0000.0000	Interrupt Control and State

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-7. Peripherals Register Map (continued)

Offset	Name	Type	Reset	Description
0xD08	VTABLE	R/W	0x0000.0000	Vector Interrupt Control and StateTable Offset
0xD0C	APINT	R/W	0xFA05.0000	Application Interrupt and Reset Control
0xD10	SYSCTRL	R/W	0x0000.0000	System Control
0xD14	CFGCTRL	R/W	0x0000.0200	Configuration and Control
0xD18	SYSPRI1	R/W	0x0000.0000	System Handler Priority 1
0xD1C	SYSPRI2	R/W	0x0000.0000	System Handler Priority 2
0xD20	SYSPRI3	R/W	0x0000.0000	System Handler Priority 3
0xD24	SYSHNDCTRL	R/W	0x0000.0000	System Handler Control and State
0xD28	FAULTSTAT	R/W1C	0x0000.0000	Configurable Fault Status
0xD2C	HFAULTSTAT	R/W1C	0x0000.0000	Hard Fault Status
0xD34	MMADDR	R/W	-	Memory Management Fault Address
0xD38	FAULTADDR	R/W	-	Bus Fault Address
Memory Protection Unit (MPU) Registers				
0xD90	MPUTYPE	RO	0x0000.0800	MPU Type
0xD94	MPUCTRL	R/W	0x0000.0000	MPU Control
0xD98	MPUNUMBER	R/W	0x0000.0000	MPU Region Number
0xD9C	MPUBASE	R/W	0x0000.0000	MPU Region Base Address
0xDA0	MPUATTR	R/W	0x0000.0000	MPU Region Attribute and Size
0xDA4	MPUBASE1	R/W	0x0000.0000	MPU Region Base Address Alias 1
0xDA8	MPUATTR1	R/W	0x0000.0000	MPU Region Attribute and Size Alias 1
0xDAC	MPUBASE2	R/W	0x0000.0000	MPU Region Base Address Alias 2
0xDB0	MPUATTR2	R/W	0x0000.0000	MPU Region Attribute and Size Alias 2
0xDB4	MPUBASE3	R/W	0x0000.0000	MPU Region Base Address Alias 3
0xDB8	MPUATTR3	R/W	0x0000.0000	MPU Region Attribute and Size Alias 3

Copyright © 2012–2019, Texas Instruments Incorporated

25.4 System Timer (SysTick) Register Descriptions

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

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

The SysTick Control and Status Register (STCTRL) register enables the SysTick features.

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

Figure 25-2. SysTick Control and Status Register (STCTRL)

31	17	16
Reserved		COUNT
R-0		R-0

15	3	2	1	0
Reserved		CLK_SRC	INTEN	ENABLE
R-0		R/W-1	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-8. SysTick Control and Status Register (STCTRL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved		Reserved
16	COUNT		Count Flag
		0	The SysTick timer has not counted to 0 since the last time this bit was read. 0
		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		Reserved
2	CLK_SRC		Clock Source
		0	External reference clock. (Not implemented for Concerto microcontrollers.)
		1	System clock Because an external reference clock is not implemented, this bit must be set in order for SysTick to operate.
1	INTEN		Interrupt Enable
		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		Enable
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

25.4.2 SysTick Reload Value Register (STRELOAD), offset 0x014

The SysTick Reload Value Register (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: This register can only be accessed from privileged mode.

Figure 25-3. SysTick Reload Value Register (STRELOAD)

31	24	23	0
Reserved		RELOAD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-9. SysTick Reload Value Register (STRELOAD) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23-0	RELOAD		Reload Value Value to load into the SysTick Current Value (STCURRENT) register when the counter reaches 0.

25.4.3 SysTick Current Value Register (STCURRENT), offset 0x018

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

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

Figure 25-4. SysTick Current Value Register (STCURRENT)

31	24	23	0
Reserved		RELOAD	
R-0		R/WC-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-10. SysTick Current Value Register (STCURRENT) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23-0	CURRRENT		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

Copyright © 2012–2019, Texas Instruments Incorporated

25.5 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 appended 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.

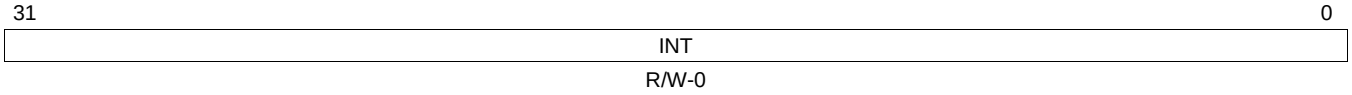
25.5.1 Interrupt 0-31 Set Enable (EN0) Register, offset 0x100

The Interrupt 0-31 Set Enable (EN0) register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. See the *Cortex-M3 Processor* chapter 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.

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

Figure 25-5. Interrupt 0-31 Set Enable (EN0) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-11. Interrupt 0-31 Set Enable (EN0) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Enable
		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 DIS*n* register.

Copyright © 2012–2019, Texas Instruments Incorporated

25.5.2 Interrupt 32-63 Set Enable 1 (EN1), offset 0x104

The Interrupt 32-63 Set Enable (EN1) register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. See the *Cortex-M3 Processor* chapter 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.

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

Figure 25-6. Interrupt 32-63 Set Enable 1 (EN1) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-12. Interrupt 32-63 Set Enable 1 (EN1) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Enable
		0	On a read, indicates the interrupt is disabled. On a write, no effect.
		1	On a read, indicates the interrupt is enabled. On a write, enables the interrupt.
			A bit can only be cleared by setting the corresponding INT[n] bit in the DIS1 register.

25.5.3 Interrupt 64-95 Set Enable 2 (EN2), offset 0x108

The Interrupt 64-95 Set Enable (EN2) register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. See the *Cortex-M3 Processor* chapter 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.

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

Figure 25-7. Interrupt 64-95 Set Enable 2 (EN2) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-13. Interrupt 64-95 Set Enable 2 (EN2) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Enable
		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 DIS2 register.

25.5.4 Interrupt 96-127 Set Enable 3 (EN3), offset 0x10C

The Interrupt 96-127 Set Enable (EN3) register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 96; bit 31 corresponds to Interrupt 127. See the *Cortex-M3 Processor* chapter for interrupt assignments.

Copyright © 2012–2019, Texas Instruments Incorporated

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.

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

Figure 25-8. Interrupt 96-127 Set Enable 3 (EN3) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-14. Interrupt 96-127 Set Enable 3 (EN3) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Enable
		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 DIS3 register.

25.5.5 Interrupt 128-133 Set Enable 4 (EN4), offset 0x110

The Interrupt 128-133 Set Enable (EN4) register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 128; bit 5 corresponds to Interrupt 133. See the *Cortex-M3 Processor* chapter 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.

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

Figure 25-9. Interrupt 128-133 Set Enable 4 (EN4) Register

31	6	5	0
Reserved		INT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-15. Interrupt 128-133 Set Enable 4 (EN4) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-0	INT		Interrupt Enable
		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 DIS3 register.

Copyright © 2012–2019, Texas Instruments Incorporated

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

The Interrupt 0-31 Clear Enable (DIS0) register disables interrupts. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-10. Interrupt 0-31 Clear Enable (DIS0) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-16. Interrupt 0-31 Clear Enable (DIS0) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Disable
		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].

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

The Interrupt 32-63 Clear Enable (DIS1) register disables interrupts. Bit 0 corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-11. Interrupt 32-63 Clear Enable (DIS1) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-17. Interrupt 32-63 Clear Enable (DIS1) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Disable
		0	On a read, indicates the interrupt is disabled. On a write, no effect.
		1	On a read, indicates the interrupt is enabled. On a write, clears the corresponding INT[n] bit in the EN1 register, disabling interrupt [n].

Copyright © 2012–2019, Texas Instruments Incorporated

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

The Interrupt 64-95 Clear Enable (DIS2) register disables interrupts. Bit 0 corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-12. Interrupt 64-95 Clear Enable (DIS2) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-18. Interrupt 64-95 Clear Enable (DIS2) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Disable
		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 EN2 register, disabling interrupt [n].

Copyright © 2012–2019, Texas Instruments Incorporated

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

The Interrupt 96-127 Clear Enable (DIS3) register disables interrupts. Bit 0 corresponds to Interrupt 96; bit 31 corresponds to Interrupt 127. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-13. Interrupt 96-127 Clear Enable (DIS3) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-19. Interrupt 96-127 Clear Enable (DIS3) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Disable
		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 EN2 register, disabling interrupt [n].

Copyright © 2012–2019, Texas Instruments Incorporated

25.5.10 Interrupt 128-133 Clear Enable (DIS4) Register, offset 0x190

The Interrupt 128-133 Clear Enable (DIS2) register disables interrupts. Bit 0 corresponds to Interrupt 128; bit 5 corresponds to Interrupt 133. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-14. Interrupt 128-133 Clear Enable (DIS4) Register

31	6	5	0
Reserved		INT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-20. Interrupt 128-133 Clear Enable (DIS4) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-0	INT		Interrupt Disable
		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 EN2 register, disabling interrupt [n].

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

The Interrupt 0-31 Set Pending (PEND0) register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-15. Interrupt 0-31 Set Pending (PEND0) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-21. Interrupt 0-31 Set Pending (PEND0) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Set Pending
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

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

The Interrupt 32-63 Set Pending (PEND1) register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-16. Interrupt 32-63 Set Pending (PEND1) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-22. Interrupt 32-63 Set Pending (PEND1) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Set Pending
		0	On a read, indicates that the interrupt is not pending. On a write, no effect.
		1	On a read, indicates that the interrupt is pending. On a write, the corresponding interrupt is set to pending even if it is disabled. If the corresponding interrupt is already pending, setting a bit has no effect. A bit can only be cleared by setting the corresponding INT[n] bit in the UNPEND1 register.

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

The Interrupt 64-95 Set Pending (PEND2) register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-17. Interrupt 64-95 Set Pending (PEND2) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-23. Interrupt 64-95 Set Pending (PEND2) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Set Pending
		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 UNPEND2 register.

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

The Interrupt 96-127 Set Pending (PEND3) register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 96; bit 31 corresponds to Interrupt 127. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 25-18. Interrupt 96-127 Set Pending (PEND3) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-24. Interrupt 96-127 Set Pending (PEND3) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Set Pending
		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 UNPEND3 register.

25.5.15 Interrupt 128-133 Set Pending (PEND4) Register, offset 0x210

The Interrupt 128-133 Set Pending (PEND4) register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 128; bit 5 corresponds to Interrupt 133. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-19. Interrupt 128-133 Set Pending (PEND4) Register

31 28	27 6	5 0
Reserved		INT
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-25. Interrupt 128-133 Set Pending (PEND4) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-0	INT		Interrupt Set Pending
		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 UNPEND2 register.

Copyright © 2012–2019, Texas Instruments Incorporated

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

The Interrupt 0-31 Clear Pending (UNPEND0) register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-20. Interrupt 0-31 Clear Pending (UNPEND0) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-26. Interrupt 0-31 Interrupt Clear Pending (UNPEND0) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Clear Pending
		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.

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

The Interrupt 32-63 Clear Pending (UNPEND1) register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-21. Interrupt 32-63 Clear Pending (UNPEND1) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-27. Interrupt 32-63 Clear Pending (UNPEND1) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Clear Pending
		0	On a read, indicates that the interrupt is not pending. On a write, no effect.
		1	On a read, indicates that the interrupt is pending. On a write, clears the corresponding INT[n] bit in the PEND1 register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt

Copyright © 2012–2019, Texas Instruments Incorporated

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

The Interrupt 64-95 Clear Pending (UNPEND2) register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-22. Interrupt 64-95 Clear Pending (UNPEND2) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-28. Interrupt 64-95 Clear Pending (UNPEND2) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Clear Pending
		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 PEND2 register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt

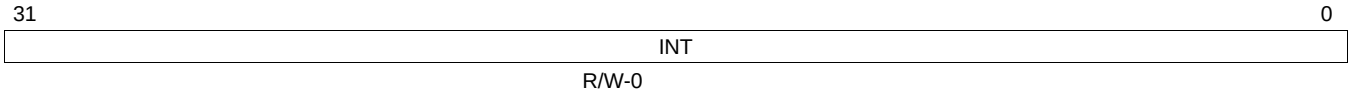
Copyright © 2012–2019, Texas Instruments Incorporated

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

The Interrupt 96-127 Clear Pending (UNPEND3) register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 96; bit 31 corresponds to Interrupt 127. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-23. Interrupt 96-127 Clear Pending (UNPEND3) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-29. Interrupt 96-127 Clear Pending (UNPEND3) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Clear Pending
		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 PEND2 register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt

Copyright © 2012–2019, Texas Instruments Incorporated

25.5.20 Interrupt 128-133 Clear Pending (UNPEND4) Register, offset 0x290

The Interrupt 128-133 Clear Pending (UNPEND4) register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 128; bit 5 corresponds to Interrupt 133. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

Figure 25-24. Interrupt 128-133 Clear Pending (UNPEND4) Register

31	6	5	0
Reserved		INT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-30. Interrupt 128-133 Clear Pending (UNPEND4) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-0	INT		Interrupt Clear Pending
		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 PEND2 register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt

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

The Interrupt 0-31 Active Bit (ACTIVE0) register indicates which interrupts are active. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

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

Figure 25-25. Interrupt 0-31 Active Bit (ACTIVE0) Register

31	0
INT	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-31. Interrupt 0-31 Active Bit (ACTIVE0) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Active
		0	The corresponding interrupt is not active.
		1	The corresponding interrupt is active, or active and pending.

Copyright © 2012–2019, Texas Instruments Incorporated

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

The Interrupt 32-63 Active Bit (ACTIVE1) register indicates which interrupts are active. Bit 0 corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

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

Figure 25-26. Interrupt 32-63 Active Bit (ACTIVE1) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-32. Interrupt 32-54 Active Bit (ACTIVE1) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Active
		0	The corresponding interrupt is not active.
		1	The corresponding interrupt is active, or active and pending.

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

The Interrupt 64-95 Active Bit (ACTIVE2) register indicates which interrupts are active. Bit 0 corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

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

Figure 25-27. Interrupt 64-95 Active Bit (ACTIVE2) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-33. Interrupt 64-95 Active Bit (ACTIVE2) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Active
		0	The corresponding interrupt is not active.
		1	The corresponding interrupt is active, or active and pending.

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

The Interrupt 96-127 Active Bit (ACTIVE3) register indicates which interrupts are active. Bit 0 corresponds to Interrupt 96; bit 31 corresponds to Interrupt 127. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

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

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 25-28. Interrupt 96-127 Active Bit (ACTIVE3) Register

31	0
INT	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-34. Interrupt 96-127 Active Bit (ACTIVE3) Register Field Descriptions

Bit	Field	Value	Description
31-0	INT		Interrupt Active
		0	The corresponding interrupt is not active.
		1	The corresponding interrupt is active, or active and pending.

25.5.25 Interrupt 128-133 Active Bit (ACTIVE4) Register, offset 0x310

The Interrupt 128-133 Active Bit (ACTIVE4) register indicates which interrupts are active. Bit 0 corresponds to Interrupt 128; bit 5 corresponds to Interrupt 133. See the *Cortex-M3 Processor* chapter for interrupt assignments.

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

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

Figure 25-29. Interrupt 128-133 Active Bit (ACTIVE4) Register

31	6	5	0
Reserved		INT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-35. Interrupt 128-133 Active Bit (ACTIVE4) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-0	INT		Interrupt Active
		0	The corresponding interrupt is not active.
		1	The corresponding interrupt is active, or active and pending.

Copyright © 2012–2019, Texas Instruments Incorporated

25.5.26 Interrupt 0-133 Priority (PRI0-PRI33) Registers, offset 0x400-0x484

The Interrupt 0-133 Priority (PRI0-PRI33) registers provide 3-bit priority fields for each interrupt. These registers are byte accessible. Each register holds four priority fields that are assigned to interrupts as follows:

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

See the *Cortex-M3 Processor* chapter 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 indicates the position of the binary point that splits the priority and subpriority fields .

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

Figure 25-30. Interrupt 0-133 Priority (PRI0-PRI33) Registers

31 29	28 24	23 21	20 16
INTD	Reserved	INTC	Reserved
R/W-0	R-0	R/W-0	R-0

15 13	12 8	7 5	4 0
INTB	Reserved	INTA	Reserved
R/W-0	R-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-36. Interrupt 0-133 Priority (PRI0-PRI33) Registers Field Descriptions

Bit	Field	Value	Description
31-29	INTD		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		Reserved
23-21	INTC		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		Reserved
15-13	INTB		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		Reserved
7-5	INTA		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		Reserved

25.5.27 Software Trigger Interrupt (SWTRIG) Register, offset 0xF00

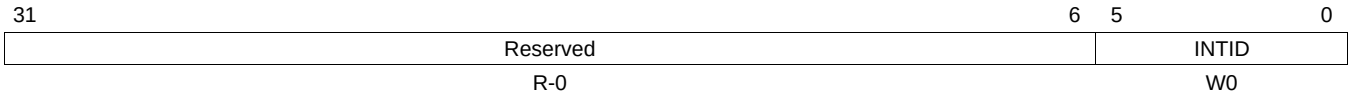
The Software Trigger Interrupt (SWTRIG) Register is described below. Writing an interrupt number to the SWTRIG register generates a Software Generated Interrupt (SGI). See the *Cortex-M3 Processor* chapter for interrupt assignments.

Copyright © 2012–2019, Texas Instruments Incorporated

When the MAINPEND bit in the Configuration and Control (CFGCTRL) register is set, unprivileged software can access the SWTRIG register.

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

Figure 25-31. Software Trigger Interrupt (SWTRIG) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-37. Software Trigger Interrupt (SWTRIG) Register Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-0	INTID		Interrupt ID This field holds the interrupt ID of the required SGI. For example, a value of 0x3 generates an interrupt on IRQ3.

Copyright © 2012–2019, Texas Instruments Incorporated

25.6 System Control Block (SCB) Register Descriptions

The System Control Block (SCB) registers are shown in numerical order by address offset. They 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.

25.6.1 Auxiliary Control (ACTLR) Register, offset 0x008

The Auxiliary Control (ACTLR) register provides disable bits for IT folding, write buffer use for accesses to the default memory map, and interruption of multi-cycle instructions. By default, this register is set to provide optimum performance from the Cortex-M3 processor and does not normally require modification.

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

Figure 25-32. Auxiliary Control (ACTLR) Register

31	3	2	1	0
Reserved		DISFOLD	DISWBUF	DISMCYC
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-38. Auxiliary Control (ACTLR) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	DISFOLD		Disable IT Folding
		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.
1	DISWBUF		Disable Write Buffer
		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-M3 processor.
0	DISMCYC		Disable Interrupts of Multiple Cycle Instructions
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

25.6.2 CPU ID Base (CPUID) Register, offset 0xD00

The CPU ID Base (CPUID) register contains the ARMA ®Cortex™-M3 processor part number, version, and implementation information.

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

Figure 25-33. CPU ID Base (CPUID) Register

31	24	23	20	19	16
IMP		VAR		CON	
R-0		R-0		R-0	

15	4	3	0
PARTNO		REV	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-39. CPU ID Base (CPUID) Register Field Descriptions

Bit	Field	Value	Description
31-24	IMP		Implementer Code
		41h	ARM
23-20	VAR		Variant NumARMber
		2h	The rn value in the rnpn product revision identifier, for example, the 2 in r2p0.
19-16	CON		Constant
		Fh	Always reads as 0xF.
15-4	PARTNO		Part Number
		0xC23	Cortex-M3 processor.
3-0	REV		Revision Number
		0h	The pn value in the rnpn product revision identifier, for example, the 0 in r2p0

Copyright © 2012–2019, Texas Instruments Incorporated

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

The Interrupt Control and State (INTCTRL) 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.

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

Figure 25-34. Interrupt Control and State (INTCTRL) Register

31	30	29	28	27	26	25	24
NMISET	Reserved		PENDSV	UNPENDSV	PENDSTSET	PENDSTCLR	Reserved
R/W-0	R-0		R/W-0	W-0	R/W-0	W-0	R-0
23	22	21		19	18		16
ISRPRE	ISRPEND	Reserved			VECPEND		
R-0	R-0	R-0			R-0		
15		12	11		10	9	8
VECPEND			RETBASE		Reserved		
R-0			R-0		R-0		
7	6						0
Reserved	V ECACT						
R-0	R-0						

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-40. Interrupt Control and State (INTCTRL) Register Field Descriptions

Bit	Field	Value	Description
31	NMISET		NMI Set Pending
		0	On a read, indicates an NMI exception is not pending. On a write, no effect
		1	On a read, indicates an NMI exception is pending. On a write, changes the NMI exception state to pending.
			Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it registers the setting of this bit, and clears this bit on entering the interrupt handler. A read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.
30-29	Reserved		Reserved
28	PENDSV		PendSV Set Pending
		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.
27	UNPENDSV		PendSV Clear Pending
		0	On a write, no effect.
		1	On a write, removes the pending state from the PendSV exception.
26	PENDSTSET		SysTick Set Pending
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-40. Interrupt Control and State (INTCTRL) Register Field Descriptions (continued)

Bit	Field	Value	Description
25	PENDSTCLR		SysTick Clear Pending
		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		Reserved
23	ISRPRE		Debug Interrupt Handling
		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		Interrupt Pending
		0	No interrupt is pending.
		1	An interrupt is pending.
			This bit provides status for all interrupts excluding NMI and Faults.
21-19	Reserved		Reserved
18-12	VECPEND		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.
		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
		...	...
		0x6B	Interrupt Vector 91
		0x6C-0x7F	Reserved
11	RETBASE		Return to Base
		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).
10-7	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-40. Interrupt Control and State (INTCTRL) Register Field Descriptions (continued)

Bit	Field	Value	Description
6-0	VECACT		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 the *Cortex-M3 Processor* chapter)

Copyright © 2012–2019, Texas Instruments Incorporated

25.6.4 Vector Table Offset (VTABLE) Register, offset 0xD08

The Vector Table Offset (VTABLE) register indicates the offset of the vector table base address from memory address 0x0000.0000.

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

Figure 25-35. Vector Table Offset (VTABLE) Register

31	30	29	28	16
Reserved		BASE	OFFSET	
R-0			R/W-0	

15	9	8	0
OFFSET		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-41. Vector Table Offset (VTABLE) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved		Reserved
29	BASE		Vector Table Base
		0	The vector table is in the code memory region
		1	The vector table is in the SRAM memory region
28-9	OFFSET		Vector Table Offset
		00h	When configuring the OFFSET field, the offset must be aligned to the number of exception entries in the vector table. Because there are 91 interrupts, the minimum alignment is 128 words.
8-0	Reserved		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

25.6.5 Application Interrupt and Reset Control (APINT) Register, offset 0xD0C

The Application Interrupt and Reset Control (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 25-42 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: This register can only be accessed from privileged mode.

Note: Determining preemption of an exception uses only the group priority field.

Table 25-42. Interrupt Priority Levels

PRIGROUP Bit Field	Binary Point[1]	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

(1) INTx field showing the binary point. An x denotes a group priority field bit, and a y denotes a subpriority field bit.

Figure 25-36. Application Interrupt and Reset Control (APINT) Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VECTKEY															
R/W-FA05h															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIANESS	Reserved				PRIGROUP			Reserved					SYSRESREQ	VECTCLRACT	VECTRESET
R-0	R-0				R/W-0			R-0					W-0	W-0	W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-43. Application Interrupt and Reset Control (APINT) Register Field Descriptions

Bit	Field	Value	Description
31-16	VECTKEY		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		Data Endianess This implementation uses only little-endian mode so this is cleared to 0.
14-11	Reserved		Reserved
10-8	PRIGROUP		Interrupt Priority Grouping This field determines the split of group priority from subpriority (see Table 25-42 for more information).
7-3	Reserved		Reserved
2	SYSRESREQ		System Reset Request 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		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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-43. Application Interrupt and Reset Control (APINT) Register Field Descriptions (continued)

Bit	Field	Value	Description
0	VECTRESET		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.

Copyright © 2012–2019, Texas Instruments Incorporated

25.6.6 System Control (SYSCTRL) Register, offset 0xD10

The System Control (SYSCTRL) register controls features of entry to and exit from low-power state.

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

Figure 25-37. System Control (SYSCTRL) Register

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		SEVONPEND	Reserved	SLEEPDEEP	SLEEPEXIT	Reserved
R-0		R/W-0	R-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-44. System Control (SYSCTRL) Register Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reserved
4	SEVONPEND		Wake Up on Pending
			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.
		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.
3	Reserved		Reserved
2	SLEEPDEEP		Deep Sleep Enable
		0	Use Sleep mode as the low power mode.
		1	Use Deep-sleep mode as the low power mode.
1	SLEEPEXIT		Sleep on ISR Exit
		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		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

25.6.7 Configuration and Control (CFGCTRL) Register, offset 0xD14

The Configuration and Control (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.

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

Figure 25-38. Configuration and Control (CFGCTRL) Register

31									16
Reserved									
R-0									
10	9	8	7	5	4	3	2	1	0
Rsvd	STKALIGN	BFHFNMIGN	Reserved		DIV0	UNALIGNED	Reserved	MAINPEND	BASETHR
R-0	R/W-1	R/W-0	R-0		R/W-0	R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-45. Configuration and Control (CFGCTRL) Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Reserved
9	STKALIGN		Stack Alignment on Exception Entry
		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		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.
		0	Data bus faults caused by load and store instructions cause a lock-up. 0
		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		Reserved
4	DIV0		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.
		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		Trap on Unaligned Access
		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		Reserved
1	MAINPEND		Allow Main Interrupt Trigger
		0	Disables unprivileged software access to the SWTRIG register.
		1	Enables unprivileged software access to the SWTRIG register.
0	BASETHR		Thread State Control
		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 the Cortex-M3 Processor chapter for more information).

Copyright © 2012–2019, Texas Instruments Incorporated

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

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.

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

Figure 25-39. System Handler Priority 1 (SYSPRI1) Register

31	24	23	21	20	16
Reserved		USAGE		Reserved	
R-0		R/W-0		R-0	

15	13	12	8	7	5	4	0
BUS		Reserved		MEM		Reserved	
R/W-0		R-0		R/W-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-46. System Handler Priority 1 (SYSPRI1) Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23-21	USAGE		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		Reserved
15-13	BUS		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		Reserved
7-5	MEM		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		Reserved

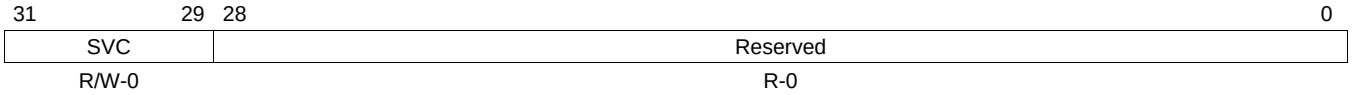
Copyright © 2012–2019, Texas Instruments Incorporated

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

The System Handler Priority 2 (SYSPRI2) register configures the priority level, 0 to 7 of the SVCall handler. This register is byte-accessible.

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

Figure 25-40. System Handler Priority 2 (SYSPRI2) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-47. System Handler Priority 2 (SYSPRI2) Register Field Descriptions

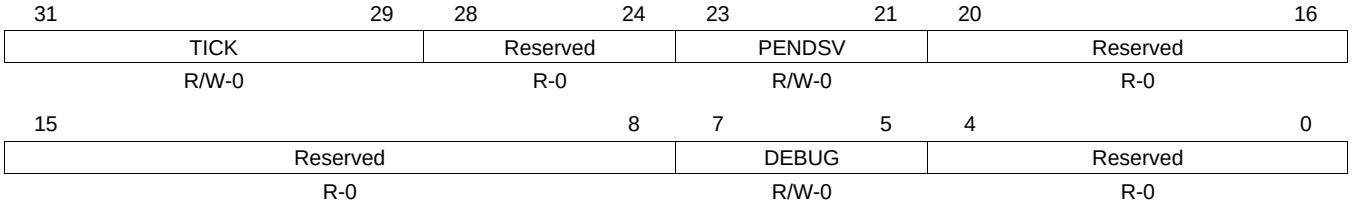
Bit	Field	Value	Description
31-29	SVC		SVCall Priority This field configures the priority level of SVCall. Configurable priority values are in the range 0-7, with lower values having higher priority
28-0	Reserved		Reserved

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

The System Handler Priority 3 (SYSPRI3) register configures the priority level, 0 to 7 of the SysTick exception and PendSV handlers. This register is byte-accessible.

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

Figure 25-41. System Handler Priority 3 (SYSPRI3) Register



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-48. System Handler Priority 3 (SYSPRI3) Register Field Descriptions

Bit	Field	Value	Description
31-29	TICK		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		Reserved
23-21	PENDSV		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		Reserved
7-5	DEBUG		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		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

25.6.11 System Handler Control and State (SYSHNDCTRL) Register, offset 0xD24

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

The System Handler Control and State (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

.

Figure 25-42. System Handler Control and State (SYSHNDCTRL) Register

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved					USAGE	BUS	MEM
R-0					R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
SVC	BUSP	MEMP	USAGEP	TICK	PNDSV	Reserved	MON
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0

7	6	5	4	3	2	1	0
SVCA	Reserved			USGA	Reserved	BUSA	MEMA
R/W-0	R-0			R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-49. System Handler Control and State (SYSHNDCTRL) Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved		Reserved
18	USAGE		Usage Fault Enable
		0	Disables the usage fault exception.
		1	Enables the usage fault exception.
17	BUS		Bus Fault Enable
		0	Disables the bus fault exception.
		1	Enables the bus fault exception.
16	MEM		Memory Management Fault Enable
		0	Disables the memory management fault exception.
		1	Enables the memory management fault exception.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-49. System Handler Control and State (SYSHNDCTRL) Register Field Descriptions (continued)

Bit	Field	Value	Description
15	SVC		SVC Call Pending
		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		Bus Fault Pending
		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		Memory Management Fault Pending
		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		Usage Fault Pending
		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		SysTick Exception Active
		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.
10	PNDSV		PendSV Exception Active
		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		Reserved
8	MON		Debug Monitor Active
		0	The Debug monitor is not active.
		1	The Debug monitor is active.
7	SVCA		SVC Call Active
		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		Reserved
3	USGA		Usage Fault Active
		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		Reserved
1	BUSA		Bus Fault Active
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-49. System Handler Control and State (SYSHNDCTRL) Register Field Descriptions (continued)

Bit	Field	Value	Description
0	MEMA		Memory Management Fault Active
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

25.6.12 Configurable Fault Status (FAULTSTAT) Register, offset 0xD28

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

The Configurable Fault Status (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:

- Read and save the Memory Management Fault Address (MMADDR) or Bus Fault Address (FAULTADDR) value.
- 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.

Figure 25-43. Configurable Fault Status (FAULTSTAT) Register

31	30	29	28	27	26	25	24
Reserved						DIV0	UNALIGN
R-0						R/W-1C-0	R/W-1C-0
23	22	21	20	19	18	17	16
Reserved				NOCP	INVPC	INVSTAT	UNDEF
R-0				R/W-1C-0	R/W-1C-0	R/W-1C-0	R/W-1C-0
15	14	13	12	11	10	9	8
BFARV	Reserved		BSTKE	BUSTKE	IMPRE	PRECISE	IBUS
R/W-1C-0	R-0		R/W-1C-0	R/W-1C-0	R/W-1C-0	R/W-1C-0	R/W-1C-0
7	6	5	4	3	2	1	0
MMARV	Reserved		MSTKE	MUSTKE	Reserved	DERR	IERR
R/W-1C-0	R-0		R/W-1C-0	R/W-1C-0	R-0	R/W-1C-0	R/W-1C-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-50. Configurable Fault Status (FAULTSTAT) Register Field Descriptions

Bit	Field	Value	Description
31-26	Reserved		Reserved
25	DIV0		Divide-by-Zero Usage Fault
		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). This bit is cleared by writing a 1 to it.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-50. Configurable Fault Status (FAULTSTAT) Register Field Descriptions (continued)

Bit	Field	Value	Description
24	UNALIGN		Unaligned Access Usage Fault
		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.
			This bit is cleared by writing a 1 to it.
23-20	Reserved		Reserved
19	NOCP		No Coprocessor Usage Fault
		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		Invalid PC Load Usage Fault
		0	A usage fault has not been caused by attempting to load an invalid PC value. 0
		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.
17	INVSTAT		Invalid State Usage Fault
		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		Undefined Instruction Usage Fault
		0	A usage fault has not been caused by an undefined instruction. The processor has attempted to execute an undefined instruction.
		1	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		Bus Fault Address Register Valid
		0	The value in the Bus Fault Address (FAULTADDR) register is not a valid fault address.
		1	The FAULTADDR register is holding a valid fault address.
			This bit is set after a bus fault, where the address is known. Other faults can clear this bit, such as a memory management fault occurring later. If a bus fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active bus fault handler whose FAULTADDR register value has been overwritten.
			This bit is cleared by writing a 1 to it.
14-13	Reserved		Reserved
12	BSTKE		Stack Bus Fault
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-50. Configurable Fault Status (FAULTSTAT) Register Field Descriptions (continued)

Bit	Field	Value	Description
11	BUSTKE		Unstack Bus Fault
		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		Imprecise Data Bus Error
		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		Precise Data Bus Error
		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.
8	IBUS		Instruction Bus Error
		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		Memory Management Fault Address Register Valid
		0	The value in the Memory Management Fault Address (MMADDR) register is not a valid fault address.
		1	The MMADDR register is holding a valid fault address.
			If a memory management fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active memory management fault handler whose MMADDR register value has been overwritten. This bit is cleared by writing a 1 to it.
6-5	Reserved		Reserved
4	MSTKE		Stack Access Violation
		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.
3	MUSTKE		Unstack Access Violation
		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		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-50. Configurable Fault Status (FAULTSTAT) Register Field Descriptions (continued)

Bit	Field	Value	Description
1	DERR		Data Access Violation
		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		Instruction Access Violation
		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.

Copyright © 2012–2019, Texas Instruments Incorporated

25.6.13 Hard Fault Status (HFAULTSTAT) Register, offset 0xD2C

The Hard Fault Status (HFAULTSTAT) register gives information about events that activate the hard fault handler. Bits are cleared by writing a 1 to them.

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

Figure 25-44. Hard Fault Status (HFAULTSTAT) Register

31	30	29	16
DBG	FORCED	Reserved	
R/W-1C	R/W-1C	R-0	

15	2	1	0
Reserved		VECT	Reserved
R-0			R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-51. Hard Fault Status (HFAULTSTAT) Register Field Descriptions

Bit	Field	Value	Description
31	DBG		Debug Event This bit is reserved for Debug use. This bit must be written as a 0, otherwise behavior is unpredictable.
30	FORCED		Forced Hard Fault
		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		Reserved
1	VECT		Vector Table Read Fault
		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		Reserved

Copyright © 2012–2019, Texas Instruments Incorporated

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

The Memory Management Fault Address (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.

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

Figure 25-45. Memory Management Fault Address (MMADDR) Register

31	0
ADDR	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-52. Memory Management Fault Address (MMADDR) Register Field Descriptions

Bit	Field	Value	Description
31	ADDR		Fault Address When the MMARV bit of MFAULTSTAT is set, this field holds the address of the location that generated the memory management fault.

25.6.15 Bus Fault Address (FAULTADDR) Register, offset 0xD38

The Bus Fault Address (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.

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

Figure 25-46. Bus Fault Address (FAULTADDR) Register Register

31	0
ADDR	
R/W	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-53. Bus Fault Address (FAULTADDR) Register Field Descriptions

Bit	Field	Value	Description
31	ADDR		Fault Address When the MMARV bit of MFAULTSTAT is set, this field holds the address of the location that generated the memory management fault.

Copyright © 2012–2019, Texas Instruments Incorporated

25.7 Memory Protection Unit (MPU) Register Descriptions

This section lists and describes the Memory Protection Unit (MPU) registers, in numerical order by address offset.

Note: The MPU registers can only be accessed from privileged mode.

25.7.1 MPU Type (MPUTYPE) Register, offset 0xD90

The MPU Type (MPUTYPE) register indicates whether the MPU is present, and if so, how many regions it supports.

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

Figure 25-47. MPU Type (MPUTYPE) Register

31	24	23	16
Reserved		IREGION	
R-0		R-0	

15	8	7	1	0
DREGION		Reserved		SEPARATE
R-0		R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-54. MPU Type (MPUTYPE) Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reserved
23-16	IREGION		Number of I Regions
		00h	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		Number of D Regions
		0x08	Indicates there are eight supported MPU data regions.
7-1	Reserved		Reserved
0	SEPARATE		Separate or Unified MPU
		0h	Indicates the MPU is unified.

Copyright © 2012–2019, Texas Instruments Incorporated

25.7.2 MPU Control (MPUCTRL) Register, offset 0xD94

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

The MPU Control (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 the *Cortex-M3 Processor* chapter. 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 the *Cortex-M3 Processor* chapter 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.

Figure 25-48. MPU Control (MPUCTRL) Register

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		PRIVDEFEN	HFNMIENA	ENABLE
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-55. MPU Control (MPUCTRL) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2	PRIVDEFEN		MPU Default Region
		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		MPU Enabled During Faults. This bit controls the operation of the MPU during hard fault, NMI, and FAULTMASK handlers.
		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

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-55. MPU Control (MPUCTRL) Register Field Descriptions (continued)

Bit	Field	Value	Description
0	ENABLE		MPU Enable. When the MPU is disabled and the HFNMIENA bit is set, the resulting behavior is unpredictable.
		0	0 The MPU is disabled.
		1	1 The MPU is enabled.

25.7.3 MPU Region Number (MPUNUMBER) Register, offset 0xD98

The MPU Region Number (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. This write updates the value of the REGION field.

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

Figure 25-49. MPU Region Number (MPUNUMBER) Register

31	3	2	0
Reserved		NUMBER	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 25-56. MPU Region Number (MPUNUMBER) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2-0	NUMBER		MPU Region to Access
		0	This field indicates the MPU region referenced by the MPUBASE and MPUATTR registers. The MPU supports eight memory regions.

25.7.4 MPU Region Base Address (MPUBASE) Register, Offset 0xD9c-0xDB4

The MPU Region Base Address (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 = Log2(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.

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

Figure 25-50. MPU Region Base Address (MPUBASE) Register

31	5	4	3	2	0
ADDR		VALID	reserved	REGION	
R/W-0		W-0	R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Copyright © 2012–2019, Texas Instruments Incorporated

Table 25-57. MPU Region Base Address (MPUBASE) Register Field Descriptions

Bit	Field	Value	Description
31-5	ADDR		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.
4	VALID		Region Number Valid
		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		Reserved
2-0	REGION		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

25.7.5 MPU Region Attribute and Size (MPUATTR) Register, offset 0xDA0-DB8

The MPU Region Attribute and Size (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 25-58 gives example SIZE values with the corresponding region size and value of N in the MPU Region Base Address (MPUBASE) register.

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

Table 25-58. Example SIZE Field Values

SIZE Encoding	Region Size	Value of N[(1)]	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

(1) Refers to the N parameter in the MPUBASE register

Copyright © 2012–2019, Texas Instruments Incorporated

Figure 25-51. MPU Region Attribute and Size (MPUATTR) Register

31	29	28	27	26	24
Reserved		XN	Reserved	AP	
R-0		R/W-0	R-0	R/W-0	

23	22	21	19	18	17	16
Reserved		TEX		S	C	B
R-0		R/W-0		R/W-0	R/W-0	R/W-0

15	8
SRD	
R/W-0	

7	6	5	1	0
Reserved		SIZE		ENABLE
R-0		R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25-59. MPU Region Attribute and Size (MPUATTR) Field Descriptions

Bit	Field	Value	Description
31-29	Reserv ed		Reserved
28	XN		Instruction Access Disable
		0	Instruction fetches are enabled.
		1	Instruction fetches are disabled.
27	Reserved		Reserved
26-24	AP		Access Privilege
		0	For information on using this bit field, see Table 25-5.
23-22	Reserved		Reserved
21-19	TEX		Type Extension Mask
		0	For information on using this bit field, see Table 25-3.
18	S		Shareable
		0	For information on using this bit field, see Table 25-3.
17	C		Cacheable
		0	For information on using this bit field, see Table 25-3.
16	B		Bufferable
		0	For information on using this bit field, see Table 25-3.
15-8	SRD		Subregion Disable Bits
		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 Section 25.2.4.1.3 for more information.
7-6	Reserved		Reserved
5-1	SIZE		Region Size Mask
		0	The SIZE field defines the size of the MPU memory region specified by the MPUNUMBER register. Refer to Table 25-58 for more information
0	ENABLE		Region Enable
		0	The region is disabled.
		1	The region is enabled.

Copyright © 2012–2019, Texas Instruments Incorporated

Revision History

Changes from March 30, 2017 to December 30, 2019 **Page**

- Table 1-12: Changed INT12 row, INTx.3 column fro Reserved to EPI. 110
- Section 1.10.3.1: Added the note at the end of the section. 149
- Table 1-157: Updated the Values column. 262
- Section 3.1: In the Intro, change "The Concerto Microcontroller" has two watchdog timer modules to The "M3 Subsystem" has two watchdog timer modules." 325
- Table 4-94: Added the note that the register is EALLOW protected. 423
- Table 6-5: Modified the table. 543
- Table 6-6: Modified the table. 544
- Section 6.5.7.9: Added this new section. 546
- Section 6.5.10.2Added bullet beginning " If boot mode is boot_to_OTP...at the end of this sectio. 548
- Figure 6-3: Added this footnote. 551
- Section 6.5.15.9: Added this new section. 570
- Table 7-52: Changed the description of 00 values. 752
- Figure 8-10: Included the new figure. 800
- Section 9.4.1.3: Deleted second paragraph already stated in Section 9.4.1.2. 831
- Table 11-8: Added the note to bit 10, ONESHOT description. 932
- Global remove References to 240x devices. 944
- Global correction for 4-level FIFOs, instead of 16. 944
- Corrected 126 to 126 different data transfer rates. 950
- Fixed SPIFFENA typo. 956
- Table 12-8: Revised the SPI SW RESET bit description. 964
- Corrected bit description regarding maximum baud rate. 967
- Fixed TXFFIVL typo. 970
- Corrected TXFFST to indicate 4-word FIFO depth. 970
- Corrected RXFFST to indicate 4-word FIFO depth. 971
- Added special note to STEINV bit description. 972
- Table 14-1: Added section after the table beginning, "To summarize..." 1012
- Chapter 19**: M3 Ethernet Media Access Controller (EMAC)** 1380
- Section 23.2.3: Revised this section and added the CAN_Muxing figure. 1516
- Table 23-2: Added a note to the bit description for Msk[28:0]. 1540

Copyright © 2012–2019, Texas Instruments Incorporated

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, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) 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.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2019, Texas Instruments Incorporated